

DESIGN OF BUFFERED ELECTRONIC CYCLIC MEMORY SYSTEMS

**A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY**

**By
OM VIKAS**

70043

to the

**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
JANUARY, 1977**

To the memory of my brother,

Late Shri Om Sharma.

LIBRARY
CENTRAL
Acc. No. **54007**

6. 1978


EE-1877-D-VIK-DES

CERTIFICATE

Certified that the work reported in the thesis entitled "Design of Buffered Electronic Cyclic Memory Systems" by Shri Om Vikas has been carried out under my supervision and has not been submitted elsewhere for the award of the degree.



Dr. V. Rajaraman
Professor
Department of Electrical Engineering
Indian Institute of Technology
KANPUR

GRADUATE OFFICE
This thesis has been approved
for the award of the Degree of
Doctor of Philosophy (Ph.D.)
in accordance with the
regulations of the Indian
Institute of Technology Kanpur
Dated: 27/12/77 

ACKNOWLEDGEMENTS

I express my deep gratitude to Professor V. Rajaraman for his invaluable guidance, insight and constant encouragement during the pursuit of this work. His affectionate words in the moments when I got dejected are unforgettable. Indeed, I consider him the builder of my career.

I would also like to thank Shri F. C. Kohli and Professor N. Deo who inspired me to join the research program.

The financial support of the Department of Electrical Engineering is highly appreciated. Many thanks are due to Shri A.K. Bhargava of the Graphic Arts for the excellent illustrations, to Shri K.N. Tewari for excellent typing and to Shri T. Tewari for neat cyclostyling. I am thankful to my friends for their help in various ways.

I express my sincere gratitude to my parents for their constant encouragement.

Finally, I wish to thank my wife, Pramuda, for her patience and understanding.

-- Om Vikas

TABLE OF CONTENTS

	Page
LIST OF TABLES	viii
LIST OF FIGURES	x
SYNOPSIS	xii
CHAPTER 1 : INTRODUCTION	1
1.1 Emergence of Gap-fillter Memories	1
1.2 Electronic Cyclic Memories	4
1.2.1 CCDs	5
1.2.2 Magnetic Bubbles	5
1.2.3 MOS Shift-Registers	6
1.2.4 Impact of ECMs	7
1.3 Buffering of ECM	7
1.3.1 Efficient Use of ECM	9
1.4 General Statement of Problems	10
1.5 An Overview of Mathematical Models for Buffer-Design	11
1.6 Summary of Dissertation	14
CHAPTER 2 : BUFFERED SERIAL MEMORY WITH CONSTANT CLOCK ECM	17
2.1 Introduction	17
2.2 Description of the Scheme	20
2.3 Operation	23
2.4 Mathematical Model	25
2.5 Design Methodology	26
2.5.1 Size of SR	26
2.5.2 Size of Buffer	26

2.6 Performance-Improvement	29
2.7 Fractional Loss of Information	31
2.8 Verification of the above Model	38
2.9 Buffer utilization	48
2.9.1 M/G/1/L with Server-Relaxation Time	48
2.9.2 G/M/1/L with Source-Relaxation Time	52
2.9.3 Verification of the Finite Queue Skinner's Model	58
2.10 Discussion of Results	58
2.11 Applications	60
2.12 Conclusion	60
CHAPTER 3 : BUFFERED STACK MEMORY WITH VARIABLE CLOCK ECM	61
3.1 Introduction	61
3.2 Description of the Scheme	65
3.3 Operation	67
3.4 Criteria of Goodness	68
3.4.1 Performance-Improvement	68
3.5 Mathematical Model	69
3.5.1 Description	69
3.5.2 Analysis	71
3.5.3 Estimation of the Gating Factors	80
3.5.4 Mean Access Time	81
3.6 Verification of the above Model	84
3.7 Discussion of Results	91
3.8 Applications	92
3.9 Conclusion	92

2.6 Performance-Improvement	29
2.7 Fractional Loss of Information	33
2.8 Verification of the above Model	38
2.9 Buffer utilization	48
2.9.1 M/G/l/L with Server-Relaxation Time	48
2.9.2 G/M/l/L with Source-Relaxation Time	52
2.9.3 Verification of the Finite Queue Skinner's Model	58
2.10 Discussion of Results	58
2.11 Applications	60
2.12 Conclusion	60
CHAPTER 3 : BUFFERED STACK MEMORY WITH VARIABLE CLOCK ECM	61
3.1 Introduction	61
3.2 Description of the Scheme	65
3.3 Operation	67
3.4 Criteria of Goodness	68
3.4.1 Performance-Improvement	68
3.5 Mathematical Model	69
3.5.1 Description	69
3.5.2 Analysis	71
3.5.3 Estimation of the Gating Factors	80
3.5.4 Mean Access Time	81
3.6 Verification of the above Model	84
3.7 Discussion of Results	91
3.8 Applications	92
3.9 Conclusion	92

CHAPTER 4 : A NOVEL TECHNIQUE FOR MULTIPLE INSERTIONS AND DELETIONS	94
4.1 Introduction	94
4.2 Description of the Scheme	95
4.3 Operation	98
4.4 Design Methodology	101
4.5 Applications	105
4.6 Conclusion	106
CHAPTER 5 : BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM	107
5.1 Introduction	107
5.2 Description of the Scheme	108
5.3 Operation	111
5.4 Mathematical Model	112
5.4.1 Mean Access Time	114
5.4.2 Performance Improvement	116
5.5 Verification of the above Model	118
5.6 Discussion of Results	124
5.7 Applications	129
5.8 Conclusion	130
CHAPTER 6 : ASSOCIATIVE ECM ORGANIZATION	131
6.1 Introduction	131
6.2 Description of the Scheme	133
6.3 Operation	135
6.4 Design Methodology	136
6.5 Applications	138
6.6 Conclusions	138

CHAPTER 7 : CONCLUSIONS AND TOPICS FOR FURTHER RESEARCH	139
7.1 Summary of Results	139
7.2 Topics for Further Research	143
LIST OF REFERENCES	146
APPENDIX A	151
A-1 Buffered Serial Memory with Constant Clock ECM	
- CDL Description	151
- Flowchart	154
A-2 Buffered Stack Memory with Variable Clock ECM	
- CDL Description	156
- Flowchart	160
A-3 A Scheme for Multiple Insertions and Deletions in an ECM	
- CDL Description	162
- Flowchart	165
A-4 Buffered Stack Memory with Constant Clock ECM	
- CDL Description	167
- Flowchart	171
APPENDIX B LISTING OF SIMULATION PROGRAMS	173

LIST OF TABLES

Table		Page
2.1	Performance-Improvement (η) of a Buffered Serial Memory with Constant Clock ECM. Analytical Values.	32
2.2	ρ in terms of f	33
2.3	Fractional Loss of Information (R_L) in Buffered Serial Memory with Constant Clock ECM. Analytical Values	36
2.4	Performance-Improvement (η) of Buffered Serial Memory with Constant Clock ECM. Comparison of Analytical and Simulation Results.	
	(a) $N = 512$	42
	(b) $N = 1024$	43
2.5	Fractional Loss of Information. Comparison of Analytical Values with Simulation Values of M/G/l/L and G/M/l/L Models	45
2.6	Heterogeneous Input and Homogeneous Output Processes	47
2.7	M/G/l/L Finite Queue Skinner's Model, $N=512$.	
	(a) $MRT = 100T_s$	54
	(b) $MRT = 200T_s$	55
	(c) $MRT = 300T_s$	56
	(d) $MRT = 400T_s$	57
3.1	Performance-Improvement (η) of Buffered Stack Memory with Variable Clock ECM	83
3.2	Buffered Stack Memory with Variable Clock ECM. Comparison of Analytical and Simulation Results.	
	(a) $N = 512$	86
	(b) $N = 1024$	87

Table	Page
3.3 Effect of Variation in Erlang Constant (E) on the Performance of a Buffered Stack with Variable Clock ECM. $N = 512$.	88
3.4 ρ in terms of f	91
4.1 Mean Service Time in a Buffered ECM. Comparison of Analytical Results with Simulation Results	103
5.1 Performance-Improvement of a Buffered Stack Memory with Constant Clock ECM. Analytical Results	117
5.2 Buffered Stack Memory with Constant Clock ECM. Comparison of Analytical and Simulation Results. (a) $N = 512$	120
(b) $N = 1024$	121
5.3 Buffered Stack Memory with Constant Clock ECM. Effect of Variation in Erlang Distribution on Performance. Simulation Results. $N = 512$.	122
5.4 Effect of Burst-Length on the Performance. of a Buffered Stack Memory with Constant Clock ECMs. Simulation Results, $N = 512$.	123
5.5 ρ in terms of f .	125

LIST OF FIGURES

Figure		Page
1.1	Price-Performance Spectrum of Memory and Storage Systems	2
1.2	Projection of Price and Performance of Memories, in 1980	3
1.3	CCDs and Magnetic Bubbles Compete with Low Capacity Disks	3
2.1	Ideal, Buffered and Unbuffered Serial Memories	19
2.2	Buffered Serial Storage with Constant Clock ECM	21
2.3	Selector-1 and Selector-2	22
2.4	Heterogeneous Input and Output Processes	27
2.5	R_L Versus L	37
2.6	η Versus L	44
2.7	Finite Queue Skinner's Models for	
	(a) M/G/1/L with Server-Relaxation Time	49
	(b) G/M/1/L with π Source-Relaxation Time	49
3.1	Ideal, Buffered and Unbuffered Stack Memories	64
3.2	Buffered Stack Memory with Variable Clock ECM	65
3.3	State-Transition-Rate Diagram	72
3.4	State-Transition Diagram	75
3.5	Transition Paths Leading to Blocking Waiting States	79

Figure	Page
3.6 Buffered Stack Memory with Variable Clock ECM - η Versus L	89
3.7 Buffered Stack Memory with Variable Clock ECM - η Versus MRT	90
4.1 ECM Organization for Non-numeric Processing	96
4.2 Selector-Unit	97
4.3 States of Buffer after (a) Initialization, (b) Deletion of a string, (c) Insertion of a string	99
4.4 \bar{x}/NT_s Versus \bar{b}	104
5.1 Buffered Stack Memory with Constant Clock ECM	109
5.2 Selector-Units	110
5.3 State Transistion Diagram	112
5.4 Buffered Stack Memory with Constant Clock ECM - η Versus L	126
5.5 Buffered Stack Memory with Constant Clock ECM - η Versus MRT	127
5.6 Effect of Burst-Length-Variation	128
6.1 (a) Basic Cyclic Memory Module (BCM) (b) An Associative ECM Organization	134
7.1 Comparison of Buffered Stack Memory with Variable and Constant Clock ECM	142

SYNOPSIS
of the
Ph.D. Dissertation
on
DESIGN OF BUFFERED ELECTRONIC CYCLIC MEMORY SYSTEMS
by
OM VIKAS

Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

January 1977

New cyclic memories, e.g., Charge-Coupled-Devices (CCDs) and Magnetic Bubbles, fill the wide gap of price and performance between semiconductor/core RAM, and magnetic disks and drums. We call these memories Electronic Cyclic Memories (ECMs) as information bits move in such memories. The class of ECMs includes dynamic and static shift-registers (SRs) as well. The clock rate is assumed to be constant in a dynamic SR, whereas it can be varied and reduced to zero in a static SR. We use the terms ECM and SR interchangeably.

Buffering of ECM will lead to a memory organization whose performance approaches that of a high speed memory and price/bit that of an unbuffered ECM. The theme of this thesis is the design of buffered ECMs which can be used with various access disciplines, namely FIFO and LIFO.

We consider an ECM of size N with a transfer rate of $1/T_s$. Requests are assumed to follow Poisson distribution. The mean inter-request time is denoted by MRT . We define the performance-improvement (η) of a buffered ECM over an unbuffered ECM as $(1 - T_b/T_u) \times 100$ percent, where T_b and T_u are the mean access times in the buffered and the unbuffered ECMs respectively.

We propose two schemes for designing buffered stack memories, one of them uses an ECM with a fixed clock rate, and the other uses an ECM with a variable clock rate. Mathematical models are developed for both the schemes. For the buffered stack memory with variable clock ECM, a queueing model is developed from first principles. Simulation study was carried out for both. Simulation results agree well with the analytical results. A performance-improvement (η) of more than 95 percent for $MRT \geq 100 T_s$ is obtained with a buffer of size 6 for a constant clock ECM of size 512 whereas a buffer of size 12 is required for a variable clock ECM of the same size.

In order to meet the requirement of a large storage with FIFO access discipline in various digital systems and to satisfy random request arrivals, we propose a buffered ECM organization with two small size buffers on input and

output ends of the ECM. ECM with constant clock rate is preferred as an additional small size shift register is required to set a condition to initiate the recirculation of information in ECM with variable clock rate.

We propose a modified Skinner's model with a finite queue to model the above scheme, and develop a computationally convenient method to analyse this model. Simulation was carried out for the above scheme. Analytical values for the performance-improvement and the fractional loss of information agree with those obtained from the simulation. With the constant clock ECM of size 512, and mean inter-insertion time and mean inter-retrieval time, $MRT \geq 100 T_s$, addition of buffers of size 6 exhibits the performance-improvement of more than 95 percent, and buffers of size 12 are required to achieve the fractional loss of insertions or retrievals less than 10^{-3} .

We propose a novel technique for multiple insertions and deletions in an ECM. A single buffer of size 16 added to an ECM yields a mean request-service time less than 0.75 cycle for mean burst-length ≤ 4 .

Finally, we propose an associative memory organization consisting of variable clock ECMs and small size buffers. The ability to vary the clock rate and the high ratio of data area to key area in a record are exploited to reduce the size of the content addressable memory (CAM) which is an expensive ingredient in this system.

We strongly feel that electronic cyclic memories, like CCDs and Magnetic bubbles will emerge as cost-effective memories in the design of computer systems. It is concluded in this thesis that with appropriate buffering of ECMs, it is possible to design a variety of useful memory systems with excellent performance characteristics.

CHAPTER 1

INTRODUCTION

1.1 EMERGENCE OF GAP-FILLER MEMORIES

Systems for information storage and retrieval, data acquisition, non-numeric processing, and stack-oriented applications, e.g., multiprocessing, subroutine operations, language translation, dynamic programming problems etc. require large size memories. Economic considerations dictate the application of cheaper storage devices, like magnetic disk and drum. However, these have very long access times. In fact there is a wide gap of price and performance between core/semiconductor random access memory (RAM) and auxiliary memory, like magnetic disk and drum. Figure 1.1 is a famous price-performance chart given by Feth (1976). Two parallel lines have been drawn through the regions of farthest advance of various technologies, roughly bracketing the price-per-bit range from the smallest memory component (say, chip) to the price-per-bit paid by the end user for a system. The difference is roughly an order of magnitude. Slope of these lines is about -0.56 , indicating that price-per-bit is proportional to $T_A^{-0.56}$, where T_A is the access time to the first bit. T_B denotes the acquisition time for a 2-K byte block. A projection of price and performance features of MOS RAM, gap-filler memories and MHD

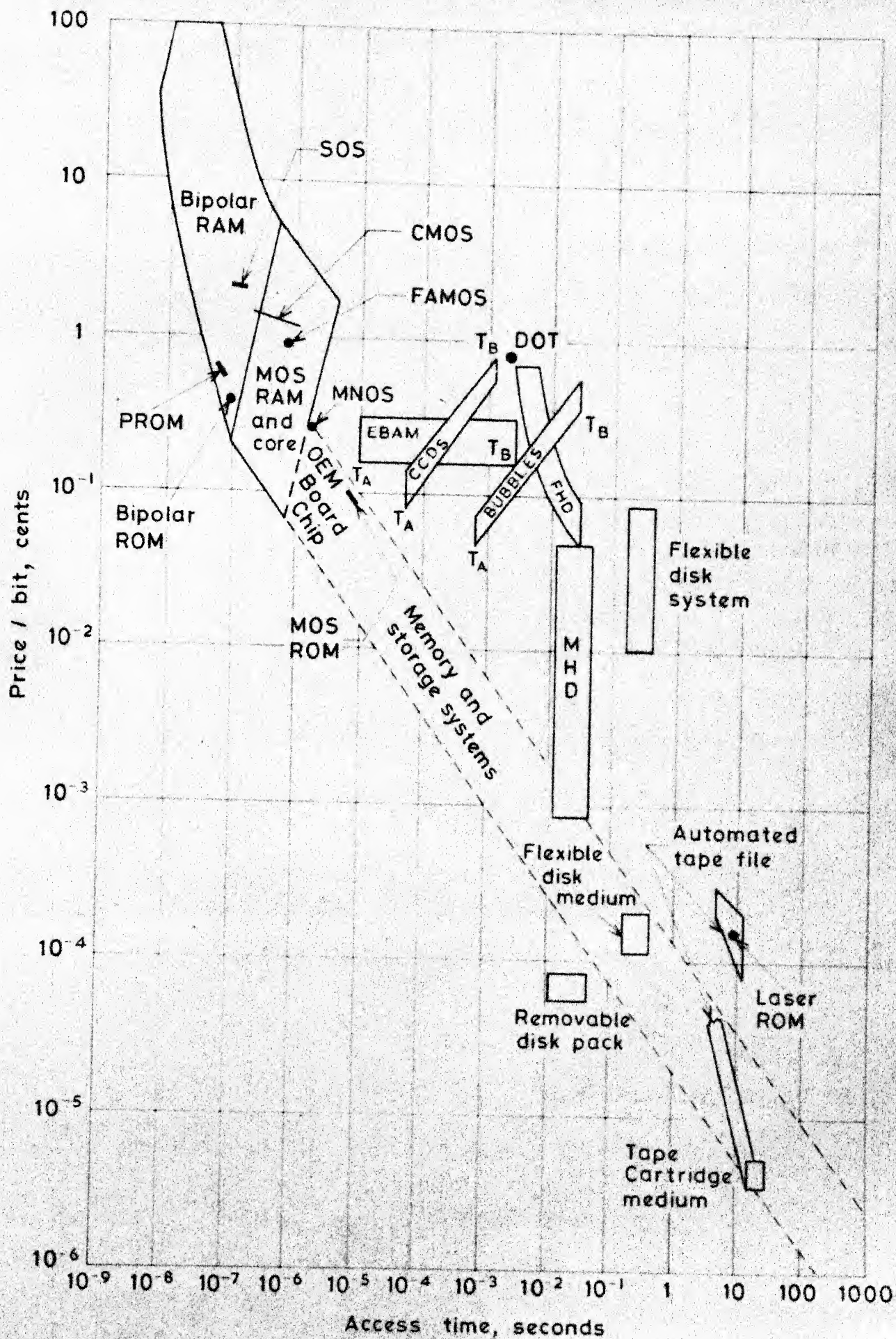


Figure 1.1 PRICE-PERFORMANCE SPECTRUM OF MEMORY AND STORAGE SYSTEMS

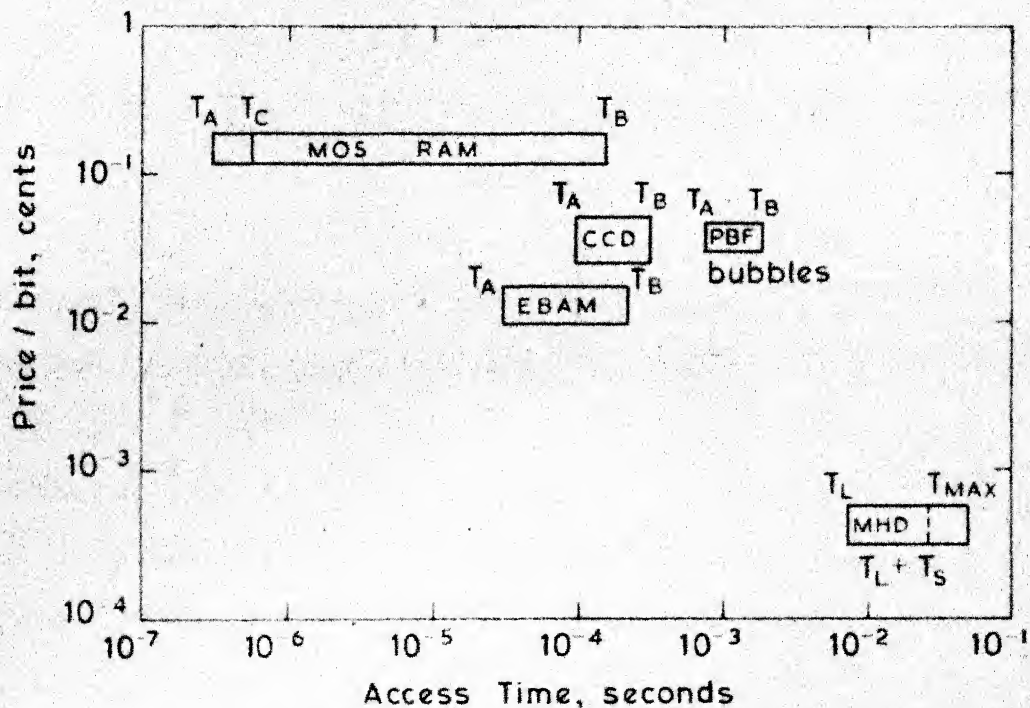


Figure 1.2 PROJECTION OF PRICE AND PERFORMANCE OF MEMORIES, IN 1980.

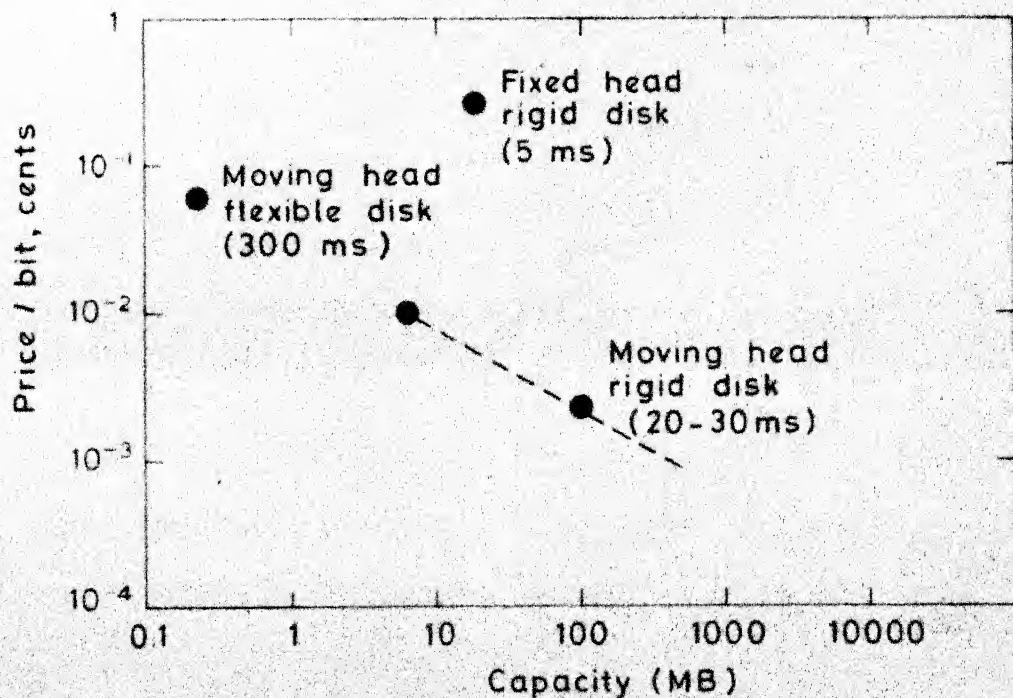


Figure 1.3 CCDs AND MAGNETIC BUBBLES COMPETE WITH LOW CAPACITY LIBRARY.

(Moving Head Disk) in 1980 is shown in Figure 1.2 [G.C. Feth, 1976].

Access time of MOS RAM (approx. 10^{-6} sec.) is less than the access time of moving head disk (approx. 10^{-2} sec.) by about four orders of magnitude. Fixed head disks are at the highest performance end of magnetic recording technology. Price-per-bit difference between MOS RAM (approx. 0.2 cents/bit) and moving head disk (approx. 0.002 cents/bit) is of two orders of magnitude. Emergence of gap-filler memories, namely, Charge Coupled Devices (CCDs), Magnetic Bubbles, Domain Tip (DOT) and Electron Beam Accessed MOS memory (EBAM or BEAMOS) fills this price-performance gap (with access time of approx. 10^{-4} sec. and price approx. 0.02 cents/bit). These are faster than mechanically rotating disk and drum, and cheaper than random-access-memory. Some of the potentially attractive features inherent in some of these technologies are low power requirements, modularity for small capacity, flexibility, higher reliability, suitability for non-numeric processing, nonvolatility and ease in maintenance.

1.2 ELECTRONIC CYCLIC MEMORIES

Information in a cyclic memory rotates cyclically past a read/write port. Cyclic memories can be classified on the basis of the mobility of information stored in them. In magnetic disks and drums, information bits remain static

on a rotating storage medium. In the case of CCDs, Magnetic bubbles and MOS shift-register memories, information bits move from one location to another in a static storage medium. This intrinsic property of mobility of information can be exploited to perform some symbol manipulative operations within memory - a further extension of distributed processing concept [Lee, 1963]. Hereafter we call a memory of the latter kind as an Electronic Cyclic Memory (ECM).

Feth [1976] presents a stimulating review of ECMs. Special issues on ECMs are listed in the bibliography.

1.2.1 CCDs

Information is stored as the quantity of charge in a potential well in the depletion layer of a MOS structure. It is shifted by applying suitably timed clock pulses. CCDs operate in a thermal non-equilibrium state, the leakage current generated tends to fill up the charge wells and thus limits the storage time. These can operate over a range of transfer rates, typically 100 KHz to 5 MHz. Since charge transfer is quite efficient, register lengths of the order of 256 bits are practical before refreshing is required. Terman and Heller [1976] give an overview of CCD Memories.

1.2.2 MAGNETIC BUBBLES

Magnetic bubbles offer data storage by remanent magnetization, as ferrite cores do. Fewer masking levels without alignment requirements in processing make the cost

per bit comparable with magnetic disk and drum. Magnetic bubbles are mobile cylindrical domains that can be generated in a film of a magnetic garnet. These bubbles can be moved around by magnetic fields which can be generated by the deposition of suitable permalloy (soft magnetic material) pattern on the film and the use of an in-plane rotating field. Interaction between currents and permalloy patterns enables one to generate, expand, contract, annihilate, split, switch or detect bubbles. Memory and logic operations can be performed as a combination of these basic operations. Writing is a controlled generation of a bubble. Reading can be done optically or by magneto-resistive detection. Bias field is generated by permanent magnets which results in nonvolatility of the magnetic bubble memory. The shift rate commonly used is 100 KHz. Higher rates, such as 1 to 10 MHz, are postulated. Salzer [1976], Cohen [1975] and Bobeck, et.al. [1975] give a good account of Magnetic Bubble Memory devices and their applications. Juliusen [1976] discusses the bubble memory systems and their advantages based on device characteristics and their unusual system-level properties.

1.2.3 MOS SHIFT-REGISTERS

There are two kinds of MOS shift-registers (SRs), namely, static and dynamic. Static MOS SR employs bistable flip-flop as a basic storage element. A current sensing

amplifier detects the signal current. In dynamic MOS-SR, information is maintained by charge storage. Charge on gate capacitance may leak through the junction capacitance which will limit effective charge-storage to a finite time period. Hence a dynamic charge-storage MOS SR memory needs periodic refreshing. Dynamic MOS SRs offer higher speed, less power dissipation and higher packing density than a static MOS SRs. Hoff and Mazor [1971] discuss the operation and the applications of MOS shift-registers from the designer's view point.

1.2.4 IMPACT OF ECMs

Wensley [1975] explores the impact of Electronic Cyclic Memories on computer architecture. He mentions, in particular, multiple minicomputers using a large central ECM, main memory acting as cache to ECM, spooling of all I/O operations by ECM, improved response of interactive data-base systems using superior access speed ECMs. Panigrahi [1976] places ECM at the third hierarchical level between MOS RAM and magnetic disk back up memory. Improvement in hierarchy performance results by virtue of disk buffering.

1.3 BUFFERING OF ECM

By a buffer memory we mean an intermediate digital store in which data is temporarily retained prior to routing to the final destination. It matches the data

rates of source(s) and acceptor(s) with an objective to minimize the loss of information. It reduces the overheads expressed in terms of CPU-idle-waiting time, interrupt process overhead, main memory cyclic stealing, etc. Finally, it synchronizes otherwise asynchronized source(s) and acceptor(s) [Philokyprou, 1975].

Gluck [1965] proposed scratchpad buffer memory consisting of CPU working registers, status registers, I/O control registers, and CPU local data buffer. This buffer memory is addressable by user programs. Wilkes [1965] proposed slave memory for 'virtual to absolute address' conversion, request queues, instruction and operand lookahead, etc. This buffer memory is invisible to user program.

Hahn [1968] classifies buffer memories into three categories, namely: (a) regularizers (i.e. derandomizers), (b) accumulators (i.e. reservoirs) and (c) synchronizers. He uses simulation as a design technique to design a buffer storage for multichannel neutron time-of-flight analysis. He makes use of multiple levels of buffering.

Philokyprou [1975] attempts to classify the buffer memories according to the service discipline, namely: (a) FIFO (First-In-First-Out), (b) LIFO (Last-In-First-Out), (c) RANDOM, (d) PRIORITY and (e) ORTHOGONAL.

1.3.1 EFFICIENT USE OF ECM

Cheaper and faster electronic cyclic memories compete with low capacity magnetic disks when the price of the disk-controller is included (Figure 1.3). A microcomputer design with ECM has been proposed by Kartashev [1976].

There are numerous applications of electronic cyclic memories with various access disciplines, namely FIFO, LIFO, etc.

This motivates us to propose efficient schemes of buffered ECM to be used with various access disciplines, and to develop analytic design techniques. The intrinsic property of the mobility of information in ECM can be exploited to develop some elegant schemes for widely used large size stacks and non-numeric applications.

We propose schemes of buffered electronic cyclic memories with FIFO and LIFO access disciplines and analyse them. Such schemes and related analytic design techniques are the main contributions of this thesis. A novel technique for insertion and retrieval of a string results in an improved stack memory organization and a scheme for symbol manipulation. Hoff and Mazor [1971] propose an elementary scheme for adding and deleting data in shift-register memory. Single bit can be inserted during a shift-period. Deletion operation results in packets of vacant positions. Our scheme enables multiple insertions during a shift period.

Multiple deletions can be performed without creating packets of vacant positions.

We propose a scheme for an inexpensive associative ECM which may be useful for interactive data-base systems. Improvement over Slotnick's logic-per-track concept [1970] results by virtue of the ability to vary clock rates in ECM. The proposed scheme will be less expensive than Slotnick's on account of reduction in the size of content addressable memory which is a very expensive ingredient.

1.4 GENERAL STATEMENT OF PROBLEMS

We propose two schemes for buffered stack memory. In one scheme, it is assumed that the clock rate of ECM can be varied and reduced to zero as found in magnetic bubbles and MOS static shift-registers. In another scheme, the clock rate of ECM is assumed to be constant. The scheme for buffered serial storage employs ECM with constant clock rate. The ability to vary the clock rate facilitates the reduction in power consumption. However, in most of the applications, the assumption of constant clock rate results in a simplified design of the logic circuitry and the better system performance.

Requests for insertion and retrieval are considered to follow Poisson distribution. Inter-request times are expressed in terms of SR shift period, T_s . Requests are assumed to be satisfied instantaneously from the buffer.

Analysis is carried out with an assumption that a single word is either inserted or retrieved.

The terms - SR and ECM, are used interchangeably. Assuming an old familiarity of the reader with shift-registers, the term, SR is frequently used in the text and on the diagrams.

1.5 AN OVERVIEW OF MATHEMATICAL MODELS FOR BUFFER-DESIGN

There is extensive literature on the behaviour of various buffer systems. In particular, N.M. Dor [1967] considers Poisson input and constant output rates. He considers an imbedded Markov chain to develop the queueing model. Given buffer-length, and input and output rates, mean information-loss due to buffer-overflow is computed. W. Chu [1970] considers Poisson input and multiple (less than or equal to a fixed number) outputs at constant rate. He computes the overflow probability as the ratio of the difference of offered load and carried load to the offered load. D.G. Maritsas and M.G. Hartley [1970] consider $E_n/D/1$ model. S. Tzafestas and G. Philokyprou [1969, 1970] consider $E_n/PD/1$ and $M/PD/1$ models. M denotes Poisson arrival, E_n Erlangian arrival, D constant service distribution, PD periodically constant service distribution, and '1' single server. They find the fractional loss of information (R_L) and the mean queue length (\bar{q}) for a buffer of length L and give the following overflow condition, $\bar{q}(1-R_L) \leq L-1$. Computation of R_L is based on Dor's work [1967]. In the above studies, the buffer is considered

word organized as each word (data-unit) is treated separately. Each word enters the buffer and then it propagates through the buffer until it joins the end of the existing queue. C. Halatsis et.al.[1974] study the statistical behaviour of the block organized buffer under Poisson input and periodic block output. Blocks are considered of fixed size (k). A buffer consists of m such blocks. Their analysis differs from Chu's [1970] only in queueing discipline, i.e. the former is for block-organized buffer whereas the latter is for word-organized buffer. A similar model is proposed by B. Powell and B. Avi-Itzhak [1967]. In this model, there is no restriction on queue length.

H. Asai and S.C. Lee [1975] consider time-varying probability distribution function of data generation, and compute the probability of congestion in a buffered system. G. Philokyprou and D.G. Maritsas [1974] consider heterogeneous input and output processes. In their model an input burst is followed by an output burst. No output occurs while an input is in progress. No new input burst can occur before an output burst has taken place. They derive a relation between buffer length and overflow probability. Their development is based on Chu's work [1970].

All these buffer designs are for FIFO service discipline. They are developed around the concept of

imbedded Markov chain. D.M.G. Wishart [1960] studies distributions of queue length and busy periods, and probability of congestion in M/G/1 queueing model with LIFO service discipline, and compares the results with those obtained in the case of FIFO service discipline. No restriction on queue length is assumed in this model.

Recently Skinner's model [1967] has been widely used in the analysis of drum storage units [S.H.Fuller and F. Baskett, 1975], and magnetic bubble memories [D.P. Bhandarkar, 1975]. Skinner [1967] proposes a queueing model with server-walking time. He considers unlimited queue length. Server inspects the queue and takes one of two decisions: if queue is empty, server becomes inactive for a variable time, and if queue is nonempty, server begins a 'service-inspection' cycle. In this model, the points in time that enjoy the Markovian property are those instants at which the server inspects the queue.

We propose a computationally convenient method for the analysis of the buffered ECM with FIFO access discipline. A modified Skinner's model is used which considers a queue of finite length in contrast with the classic Skinner's model [1967] with no restriction on queue length. U.N.Bhat [1973] suggests computationally convenient analytic methods for the analysis of M/G/1 and G/M/s

finite queueing systems, where G denotes general distribution and s the number of servers.

We develop an $M/M/1$ model from first principles for the analysis of the buffered variable clock ECM (e.g., static SR) with LIFO access discipline.

1.6 SUMMARY OF DISSERTATION

This dissertation presents the schemes of buffered electronic cyclic memories with various access disciplines, namely, FIFO and LIFO. A novel technique for insertion and deletion in ECM is developed. This dissertation also analyzes these schemes so as to provide a designer with the analytical tools to prepare a design guide table. A brief outline of this dissertation is given below.

Following this introduction, Chapter 2 presents the design of a buffered serial memory consisting of a large size electronic cyclic memory and two small size buffers on both ends of the cyclic memory to cater to the input and output requirements. The design criteria to determine the buffer-length are the fractional loss of information on input and output ends and the improvement in performance over an unbuffered ECM.

The input and the output buffers are analysed independently. Interaction between a buffer and the ECM takes place after a fixed period of time as the information is assumed to

recirculate at a fixed transfer-rate. A finite queue Skinner's model is used to obtain the utilization of buffer. Simulation is conducted to verify the analytical results. It is concluded in this chapter that by adding buffers of size 6 bits to an ECM of 512 bits, the terminal performance of the resultant memory tends to that of ideal FIFO memory of length 512 with 95 percent performance-improvement for $MRT \geq 100 T_s$.

Chapter 3 discusses the design of a buffered stack memory consisting of an ECM with variable clock rate and a small size buffer (fast-stack). Analytic, as well as simulation, results are presented to quantify the performance of the stack memory organization. Measure of performance is considered the improvement in access time over an entirely unidirectional SR. A mathematical model is developed from first principles. It is concluded in this chapter that by adding a buffer of 12 bits to an ECM of 512 bits, the terminal performance of the resultant stack memory tends to that of ideal stack memory of length 512 with 95 percent performance improvement for $MRT \geq 100 T_s$.

Starting with Chapter 4, we investigate the possibility of exploiting the property of the mobility of information in ECM. Chapter 4 reports a novel technique for insertion and deletion of a string in an ECM. A small size buffer is employed to perform these operations. Such a scheme is useful for non-numeric processing.

Chapter 5 presents an interesting design of a buffered stack memory consisting of a dynamic SR and a small size buffer. Buffer is periodically initialized to half of its capacity. Techniques of insertion and deletion in an ECM (Chapter 4) are employed, to perform the initialization. Analytic, as well as simulation, results are presented to quantify the performance of the scheme. Improvement (in access time) over an entirely unidirectional SR is considered as the measure of performance. Strikingly, the improvement rises faster with an increase in buffer-length(cf.Chapter 3). It is concluded in this chapter that by adding a buffer of size 6 bits to an ECM of length 512, the terminal performance of the resultant stack memory organization tends to that of ideal stack memory with more than 95 percent performance-improvement for $MRT \geq 100 T_s$.

Chapter 6 describes a conceptual organization for an associative electronic cyclic memory. Ability to vary clock rate and the knowledge that the ratio of data area to key area is large are employed to reduce the size of Content Addressable Memory. An associative operation is performed in two phases, namely search and follow-up operations. Addition of a set of parallel-in-serial-out shift-registers enables one to perform string manipulative operations. Multiple (sequential) search operations over a word during a single shift period enhances the parallelism.

Chapter 7 summarizes the work of this dissertation

CHAPTER 2

BUFFERED SERIAL MEMORY WITH CONSTANT CLOCK ECM

A scheme is proposed for a buffered serial memory consisting of a large size ECM and two small size buffers on both of the ends of the ECM. In order to determine the optimal length of buffers, two design criteria, namely the performance-improvement over an unbuffered FIFO storage and the fractional loss of information due to buffer-overflow are considered.

2.1 INTRODUCTION

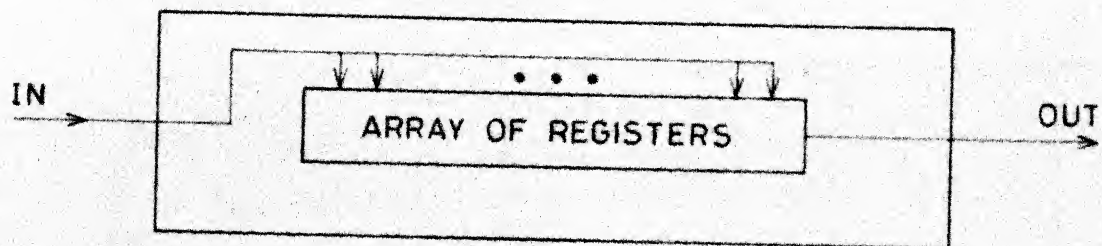
A First-In-First-Out (FIFO) serial storage is used in many areas of digital system design. It serves as a reservoir when the retrieval rate is less than the average insertion rate over a (large) period of time. Electronic cyclic memories, e.g., MOS shift-register, CCDs and magnetic bubbles form a FIFO serial storage. We use the terms Electronic Cyclic Memory (ECM), and Shift-Register (SR) interchangeably.

Insertion and retrieval request patterns to the storage are assumed to be independent random events. If insertion is made directly into SR at the instant of its arrival, packets of holes (vacant positions) will be created within SR resulting in the problem of identification of first data entry and of garbage collection as data is scattered over the entire length of SR. A retrieval request

cannot be satisfied until the first data entry is detected. Consecutive insertions will, in turn, facilitate consecutive retrieval of data.

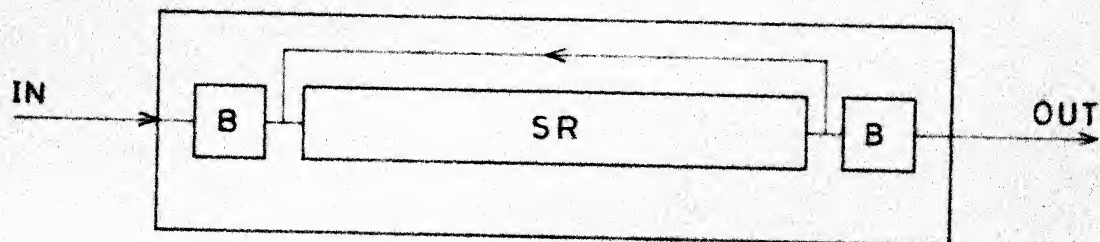
Hahn proposed a multi-level buffer storage scheme for neutron time-of-flight analysis [Hahn, 1968]. A typical three level buffer storage system by Hahn includes ultra high speed, high speed, and slow speed buffers. Internal structure of buffers is not discussed. 'Insertion into' takes higher priority over 'retrieval from' the high speed buffer.

We propose a scheme to realize a FIFO storage consisting of a large size unidirectional shift-register memory (ECM) and two small size buffers on input and output ends. Buffers on input and output ends facilitate, independently, fast and simultaneous insertion and retrieval respectively. The input buffer is emptied into SR and similarly, the output buffer is filled from SR either periodically or at the instants determined by some condition. Once the tail, that is the last entry, of the information-chunk (ic) leaves the input port, it can be back after an interval equal to N or more shift-periods. The same holds for the head, that is the first entry, of ic. The length of the buffers should be large enough so that the performance of the buffered FIFO storage (Figure 2.1(b)) approaches that of an ideal FIFO storage



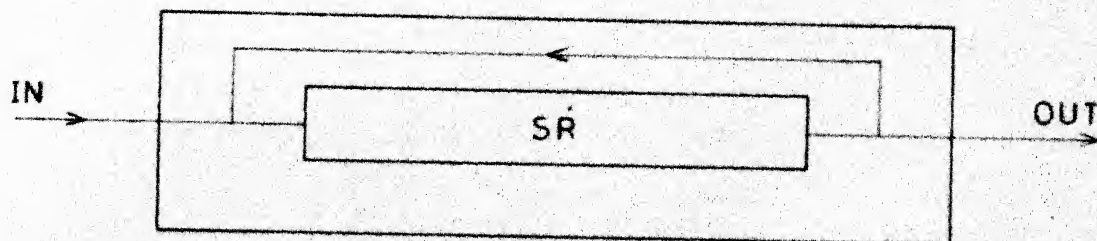
(a)

IDEAL
FIFO STORAGE



(b)

BUFFERED
FIFO STORAGE



(c)

UNIDIRECTIONAL
SR FIFO STORAGE

Figure 2.1 SERIAL MEMORY ORGANIZATIONS.

consisting of an array of registers (Figure 2.1(a)) and its price/bit tends to that of the entirely unidirectional SR FIFO storage (Figure 2.1(c)).

If the clock-rate is constant, the tail and the head of it reach the input and the output ports respectively at a fixed interval of the recirculation period.

The ability to vary the clock rate of an ECM, like static shift-registers and CCDs appear an attractive property. However, this will require a third buffer in order to provide a condition to initiate the recirculation of SR. This scheme may be desirable where reduction in power-dissipation is prime importance. Consideration of constant clock-rate simplifies the logic circuitry and only two buffers suffice. In the scheme mentioned here-below, we consider ECM with a constant clock rate, e.g., dynamic shift-registers.

2.2 DESCRIPTION OF THE SCHEME

m shift-registers, each of N -bits, constitute a serial storage of N -words, m -bits/word. Figure 2.2 depicts a configuration of a single unidirectional dynamic shift-register (SR) with two buffers, namely B_1 and B_2 on input and output ends. An additional buffer of the same length can be used to store the activity bits for m buffers on the input end and similarly one on the output end. An activity bit is 0 for vacant position and 1 for non-vacant position. Use of such a buffer simplifies the design of the selector-unit (see Figure 2.3).

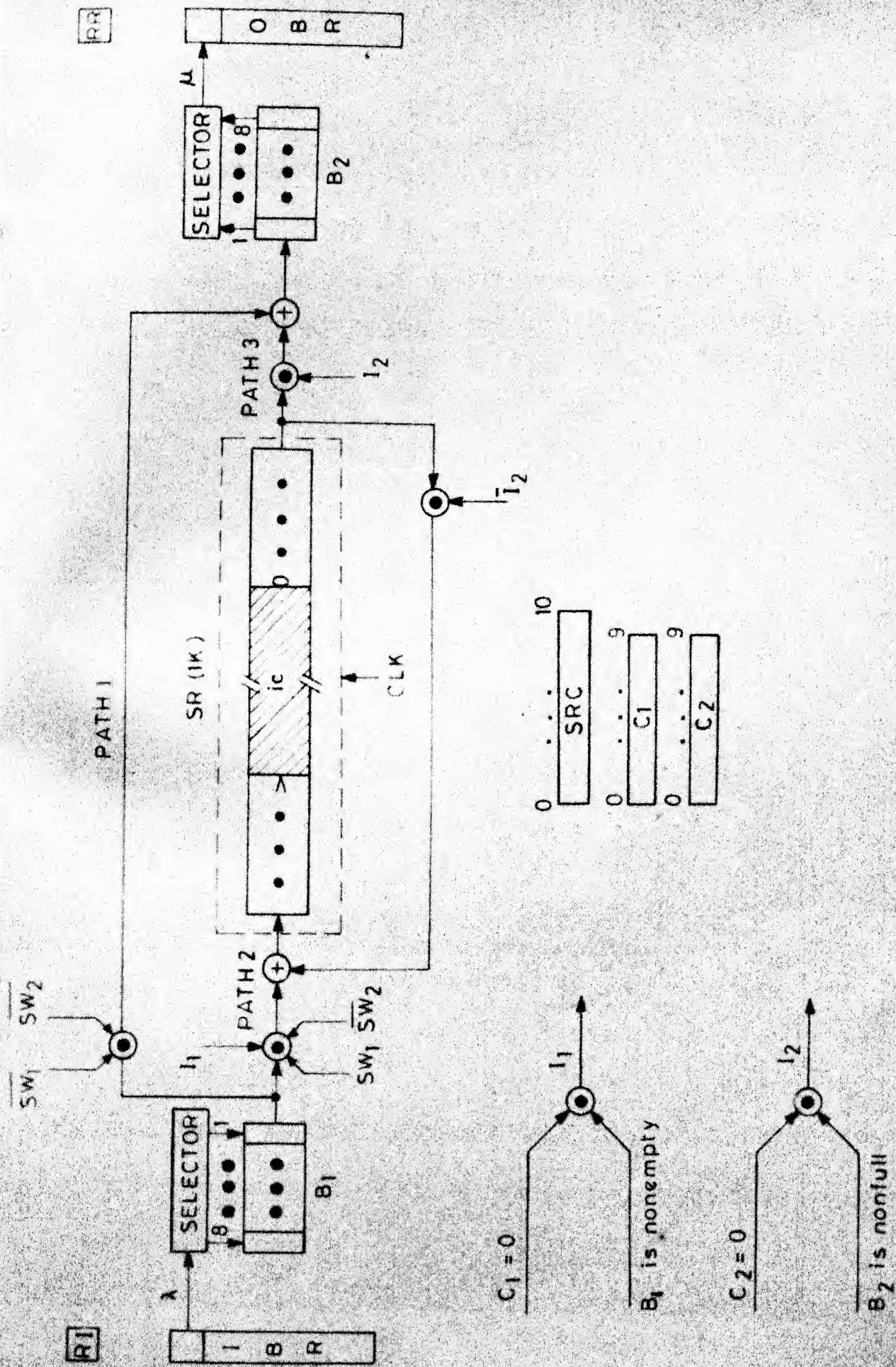
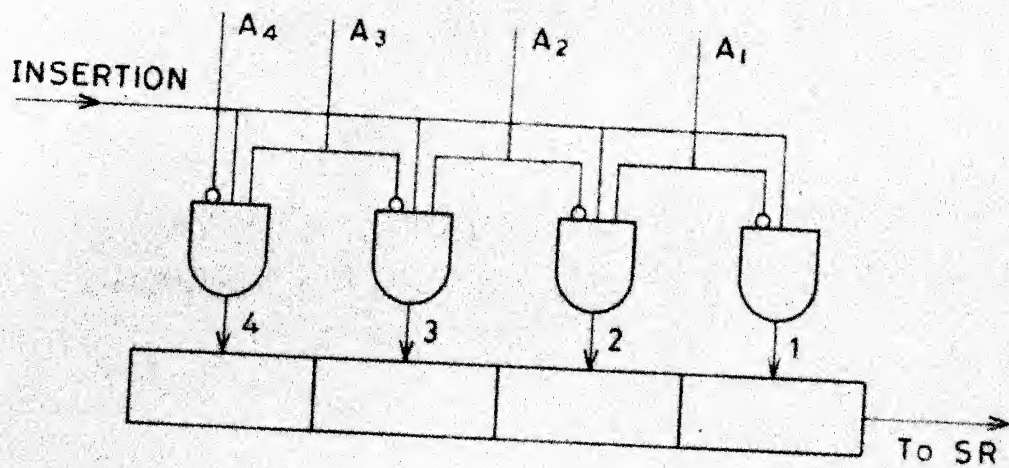
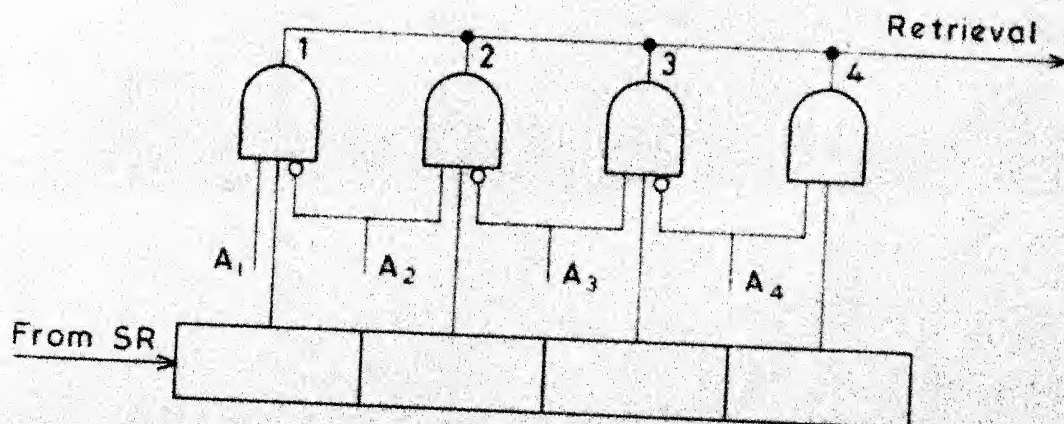


FIG. 2.2 BUFFERED SERIAL STORAGE
WITH CONSTANT CLOCK ECM



(a) SELECTOR-1

sets lowest index pointing
to vacant position.



(b) SELECTOR-2

sets highest index pointing
to non-vacant position.

Figure 2.3 SELECTOR UNITS.

The input buffer is, essentially, a Parallel-In-Serial-Out (PISO) shift-register (typically of size 8-32 bits). The output buffer is a Serial-In-Parallel-Out (SIPO) shift-register. A selector unit is associated with each buffer. Selector-1 is a combinatorial circuit to activate a switch in order to store an input-data into the rightmost vacant position of B_1 . Selector-2 makes the rightmost data location of B_2 available for instant retrieval. Counter C_1 indicates the distance of the tail, that is the last entry, of the information-chunk (ic) from the input port of SR. Counter C_2 indicates the distance of the head, that is the first entry, of ic from the output port of SR. Distance is measured in terms of number of shifts. Each shift-clock increments the distance by one. The size of ic can be obtained by the magnitude difference between C_1 and C_2 .

Flip-flop SW_1 is reset when SR becomes empty and set when B_2 becomes full while SR is still empty. Flip-flop SW_2 is set when SR becomes full and remains reset otherwise.

2.3 OPERATION

Let N be the size of SR and $1/T_s$ be its shift-rate. Further, we assume the buffers of equal length, L . Figure 2.2 depicts a configuration with $N=1024$ and $L=8$.

As a request for insertion arrives, the occupancy of B_1 is examined. If B_1 is found full the request is

rejected otherwise an insertion is made into the rightmost vacant position of B_1 which is selected by SELECTOR-1. On the other hand, a request for retrieval is rejected if B_2 is found empty otherwise the rightmost data word stored in B_2 is retrieved through SELECTOR-2 and the corresponding location in B_2 is left as a vacant one after decrementing the switch index by one.

Path₁ (see Figure 2.) is activated whenever SR becomes empty. This connects B_1 , directly to B_2 and data from B_1 is immediately transferred to B_2 until B_2 gets full. As SR is empty, SW_1 will be set when B_2 gets full. Path₂ is activated whenever SW_1 is set and the tail of the information-chunk (ic) reaches the input port of SR. Now, data from B_1 is transferred consecutively into SR until B_1 is left empty. Path₃ is activated whenever the head of ic reaches the output port of SR and then data from SR is transferred into B_2 until B_2 gets full. Rest of the data stored in SR keeps recirculating.

Counter C_1 is incremented at every shift operation during the entire cycle except when Path₂ remains activated. C_1 stops counting up during the period of consecutive transfers from B_1 into SR. It is to be noted that such a transfer was initiated when $C_1 = 0$, hence C_1 remains at zero during the activation-period of Path₂. Similarly, counter C_2 is incremented at every shift operation during the entire cycle except when Path₃ remains activated. C_2 remains at zero during the activation-period of Path₃.

2.4 MATHEMATICAL MODEL

Requests for insertion and retrieval are assumed to follow Poisson distribution which is the simplest arrival process facilitating mathematical analysis. Insertion- and retrieval requests are equally likely events. The Poisson process has been widely studied [W.Feller, 1968]. The fundamental properties of a Poisson arrival process are: any two time intervals of equal length experience an arrival with equal probability, and the number of arrivals during disjoint intervals are independent random events. Inter-arrival times have the negative exponential density function $\lambda e^{-\lambda t}$, where λ is the mean arrival rate, and the probability of k arrivals during an arbitrary interval of time, T , is $\frac{(\lambda T)^k}{k!} e^{-\lambda T}$. Poisson assumption facilitates analysis, and under certain conditions may be a reasonable approximation to reality.

The information-chunk (ic) recirculates at a constant rate within SR. As the tail of ic reaches the input port of SR, the input end buffer is inspected and emptied into SR during consecutive shift periods. The tail leaves the buffer when it becomes empty and inspects it again after a fixed period of time, $N.T_s$, which we call the relaxation period. Similarly, as the head of ic reaches the output of SR, the output end buffer is inspected and filled from SR during consecutive shift periods. The head leaves the buffer when it becomes full and inspects it again after the relaxation period, $N.T_s$.

A request for insertion is rejected when the input-buffer is found full. Similarly, a request for retrieval is rejected when the output buffer is found empty.

2.5 DESIGN METHODOLOGY

2.5.1 SIZE OF SR

Requirement for a large size shift-register is felt in the case of heterogeneous input and output processes. Parameters in a homogeneous process remain constant in time, whereas they vary with time in a heterogeneous process [G.Philokyprou, 1974].

Let λ_i and μ_i be the mean request rates for insertion and retrieval respectively over an interval of time t_i (see Figure 2.4). We may, possibly, define a period of insertion and retrieval patterns that repeats. Let such a period be T consisting of k subintervals.

$$T = \sum_{i=1}^k t_i$$

For heterogeneous input and output processes,

$$N > \max (Z_j) \quad \text{for } j = 1, 2, \dots, k$$

where
$$Z_j = \sum_{i=1}^j (\lambda_i - \mu_i) t_i$$

2.5.2 SIZE OF BUFFER

In order to determine the optimal size of a buffer on input or output end, we define two measures of performance, namely mean access time (T_b) and fractional loss of

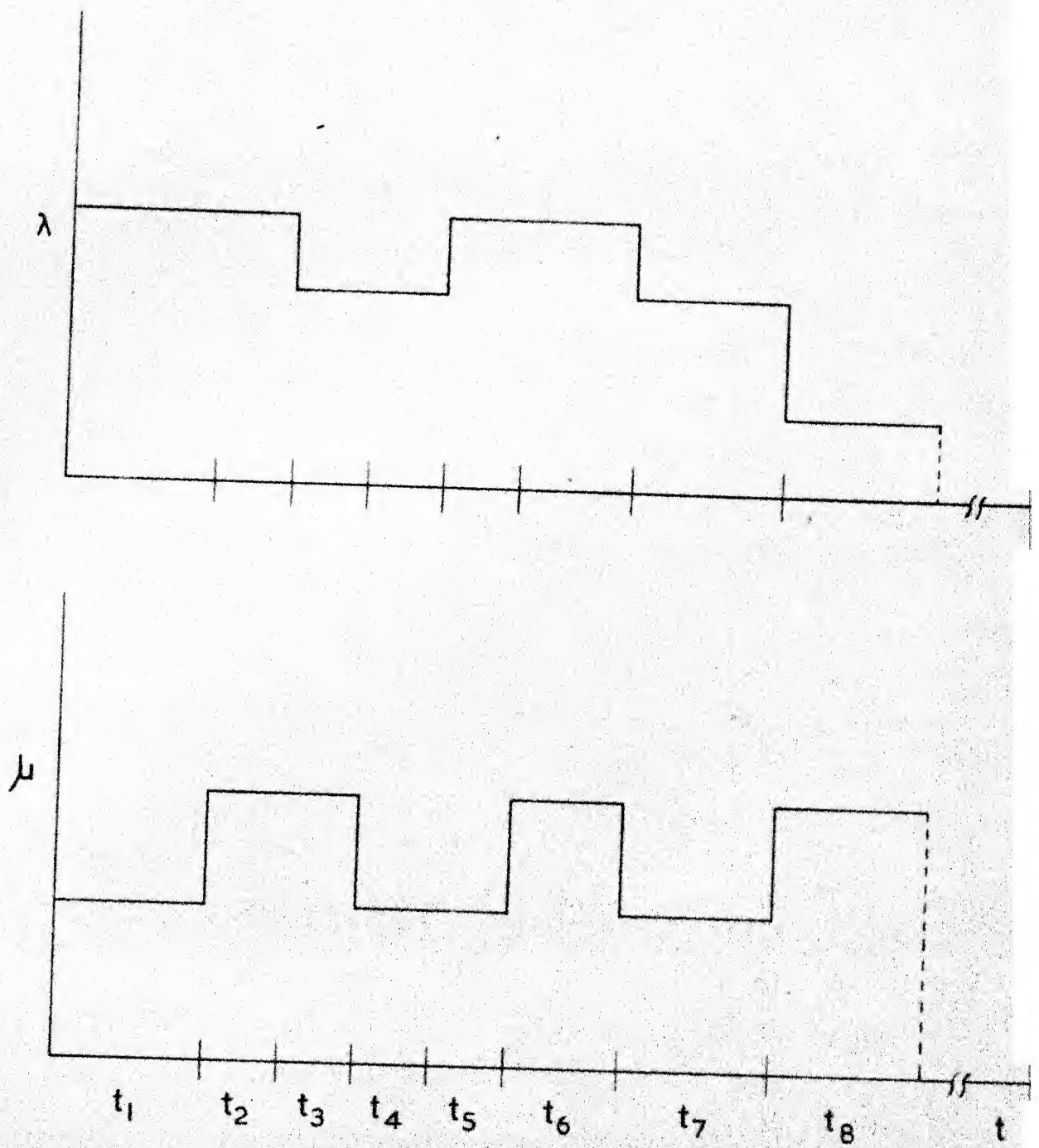


Figure 2.4 HETEROGENEOUS INPUT AND OUTPUT PROCESSES

information (R_L). To illustrate with an example, we consider the input buffer. Requests for insertion are satisfied instantaneously until the input buffer gets full. In the case where a request for insertion is required to wait when the buffer is full, the mean access time is a suitable measure of performance. For a buffer of length L , the mean access time (T_b) is computed as the ratio of waiting time due to $(L+1)$ -th request to the expected number of requests serviced during the relaxation period and the period of data-transfer from buffer to SR. Further, we define performance-improvement (η) as $(1 - T_b/T_u) \times 100$ percent, where T_u is the mean access time of an unbuffered serial storage.

There are many applications in which the fractional loss of information is a suitable measure of performance. In this case, requests arriving after the buffer gets full are lost. Unlike the previous case, the generation of requests is not affected by the occurrence of buffer-overflow. The fractional loss of information (insertion-requests) is computed as the ratio of the number of requests lost to the total number of requests during the relaxation period and the period of data-transfer between buffer and SR.

Input and output buffers interact with each other only when Path 1 is activated which is a rare event. Hence both the buffers are assumed to function independently. A

buffer is inspected and information is transferred during consecutive shift-periods until the condition to initiate the period of inactivity (relaxation) is reached. The relaxation period starts with a fixed initial condition and lasts for a fixed period of time. We, therefore, study the requests arriving at a buffer during the relaxation period.

Design-guide tables are prepared for both η and R_L . Given N and MRT , one can obtain an optimal buffer-length from these tables so as to achieve η greater than a specified value and R_L less than a specified ratio.

2.6 PERFORMANCE-IMPROVEMENT

An ideal FIFO storage is an array of registers (refer to Figure 2.1(a)). Such a storage offers a zero mean access time. On the contrary, a FIFO storage consisting of entirely unidirectional SRs (Figure 2.1(c)) suffers from a large mean access time, that is $N \cdot T_s / 2$ where N is the size of SR with a transfer-rate of $1/T_s$. A buffered FIFO storage (Figure 2.1(b)) shows the mean access time between these two extreme values. We define the performance-improvement (η) of a buffered FIFO storage over an unbuffered SR FIFO storage as follows.

$$\eta = \left(1 - \frac{T_b}{T_u}\right) \times 100 \text{ percent}$$

where T_b = mean access time of the buffered FIFO storage,
and T_u = mean access time of the unbuffered SR FIFO storage.

In the case of an ideal FIFO storage, $T_b = 0$ and hence $\eta = 100$ percent. For an unbuffered SR FIFO storage, $T_b = T_u$ and hence $\eta = 0$ percent, that means no improvement.

Probability distribution function of the time, t , for the n -th request is computed from the n th convolution of probability density function of inter-request intervals.

$$H_n(t \leq \tau) = \int_0^{\tau} \frac{\lambda(\lambda t)^{n-1}}{(n-1)!} e^{-\lambda t} dt$$

Mean waiting time of the n -th request during the relaxation period, $N.T_s$,

$$\bar{W}_n = \int_0^{NT_s} \frac{\lambda(\lambda t)^{n-1}}{(n-1)!} e^{-\lambda t} dt - \frac{n}{\lambda} \int_0^{NT_s} \frac{\lambda(\lambda t)^n}{n!} e^{-\lambda t} dt$$

In our example, $(L+1)$ -th request waits. Hence,

$$\begin{aligned} \bar{W}_{L+1} = & NT_s \left[1 - \sum_{j=0}^L \frac{(\lambda NT_s)^j}{j!} e^{-\lambda NT_s} \right] \\ & - \frac{(L+1)}{\lambda} \left[1 - \sum_{j=0}^{L+1} \frac{(\lambda NT_s)^j}{j!} e^{-\lambda NT_s} \right] \end{aligned}$$

[H.M. Wagner, 1973]

Let \bar{r} be the expected number of requests being satisfied during the relaxation period and the period of data-transfer from buffer to SR. Probability of buffer-overflow, P_{of} , can be computed as follows.

$$P_{of} = \left[1 - \sum_{j=0}^L \frac{(\lambda NT_s)^j}{j!} e^{-\lambda NT_s} \right]$$

We can write

$$\bar{r} = (L+1) + P_{of} \sum_{k=1}^{\infty} \frac{k[\lambda(L+k-1)T_s]^k}{k!} \exp[-\lambda(L+k-1)T_s]$$

Hence the mean access time,

$$T_b = \bar{W}_{L+1} / \bar{r}$$

and the performance-improvement,

$$\eta = (1 - T_b/T_u) \times 100 \text{ percent.}$$

The analytical values for the performance-improvement (η) are recorded in Table 2.1 for $N = 512, 1024$ and 2048 . $L = 2, 4, 6, 8, 12, 16$ and 24 ; and $MRT/T_s = 25, 50, 75, 100, 150, 200, 250, 300, 350$ and 400 . To illustrate with an example, η for $L = 8$ and $MRT = 100 T_s$ can be obtained from this table as 99.7 percent for $N = 512$ and 95.8 percent for $N = 1024$.

It is to be noted that η depends on the product $\rho = \lambda N T_s$. The values for η are fairly close within an accuracy of 0.5 percent for a specified value of ρ obtained from different pairs of N and MRT . We define a constant f equal to $10.24 \cdot \rho$ can be expressed in terms of f , e.g., $\rho = 5.12 = f/2$ for ($N = 512, MRT = 100$) as well as ($N = 1024, MRT = 200$). is tabulated in Table 2.2 for different pairs of N and MRT .

TABLE 2.1 Performance-improvement (η) of a buffered
serial memory with constant clock ECM.
Analytical values.

N = 512										
L	MRT ($\times T_s$) \longrightarrow									
	25	50	75	100	150	200	250	300	350	400
2	43.9	53.2	62.4	70.4	81.9	88.5	92.4	94.7	96.2	97.2
4	70.3	89.6	87.5	92.6	97.4	98.9	99.5	99.8	99.9	99.9
6	81.5	90.4	96.0	98.4	99.7	99.9	100.0	100.0	100.0	100.0
8	87.8	95.8	98.9	99.7	100.0	100.0				
12	94.4	99.4	100.0	100.0						
16	97.7	100.0								
24	99.8	100.0								
N = 1024										
2	39.1	43.5	48.2	53.0	62.3	70.4	76.9	81.9	85.6	88.5
4	65.5	70.0	74.8	79.5	87.4	92.6	95.6	97.4	98.3	98.9
6	76.7	81.4	86.1	90.4	96.0	98.4	99.3	99.7	99.9	99.9
8	83.0	87.7	92.2	95.8	98.9	99.7	99.9	100.0	100.0	100.0
12	89.7	94.4	98.0	99.4	100.0	100.0	100.0			
16	93.2	97.7	99.7	100.0						
24	96.9	99.8	100.0							
N = 2048										
2	36.7	38.7	40.9	43.3	48.1	52.9	57.7	62.2	66.5	77.3
4	63.1	65.2	67.5	69.9	74.7	79.4	83.7	87.4	90.3	92.6
6	74.4	76.5	78.9	81.3	86.1	90.4	93.7	96.0	97.5	98.4
8	80.6	82.8	85.2	87.6	92.2	95.8	97.9	98.9	99.5	99.7
12	87.3	89.6	92.0	94.4	98.0	99.4	99.9	100.0		
16	90.8	93.2	95.6	97.7	99.7	100.0				
24	94.5	96.9	99.0	99.8	100.0					

TABLE 2.2

 ρ in terms of f .

$\begin{array}{c} \text{MRT}(x T_s) \\ N \end{array}$	25	50	75	100	150	200	300	400
512	$2f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$	$\frac{1}{3}f$	$\frac{1}{4}f$	$\frac{1}{6}f$	$\frac{1}{8}f$
1024	$4f$	$2f$	$\frac{4}{3}f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$	$\frac{1}{3}f$	$\frac{1}{4}f$
2048	$8f$	$4f$	$\frac{8}{3}f$	$2f$	$\frac{4}{3}f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$

2.7 FRACTIONAL LOSS OF INFORMATION

In order to determine the fractional loss of information, we define two terms, namely, offered load (OL) and carried load (CL). OL is the expected number of requests arriving during the relaxation period, $N \cdot T_s$, and the period of data transfer between buffer and SR. CL is the expected number of requests which are satisfied. We can define the fractional loss of information, say R_L , as follows:

$$R_L = (OL - CL)/OL$$

It is to be noticed that the probability of two or more requests arriving during a shift-period, T_s , is less than 10^{-4} for mean inter-arrival times $100 \cdot T_s$ and more. Therefore, we can consider that all the requests arriving during the period of data transfer between buffer and SR are satisfied and hence there is the loss of information. Further we can write ,

$$OL = (OL)_1 + (OL)_2$$

$$CL = (CL)_1 + (CL)_2$$

where $(OL)_1$ and $(CL)_1$ are the offered and the carried loads during the relaxation period, $N.T_s$. $(OL)_2$ and $(CL)_2$ are the offered and the carried loads during the period of data-transfer between buffer and SR. From the assumption that there is no loss of information during this period, $(OL)_2 = (CL)_2$.

Let λ be the mean arrival-rate of requests for insertion and L be the length of buffer B_1 . Relaxation period is initiated at the instant B_1 gets empty. Insertions during the relaxation period change the occupancy of the buffer. Occupancy of the buffer corresponds to the carried load. We define,

$b_i = \text{Prob}[i \text{ items in the buffer, } B_1, \text{ at the end of the relaxation period}]$.

$$b_i = \frac{(\lambda NT_s)^i}{i!} \exp(-\lambda NT_s) \quad \text{for } i = 0, 1, \dots, (L-1)$$

and

$$B_L = \sum_{k=L}^{\infty} \frac{(\lambda NT_s)^k}{k!} \exp(-\lambda NT_s)$$

$$(OL)_1 = \lambda NT_s.$$

$$(OL)_2 = \sum_{k=0}^{\infty} \sum_{i=1}^L b_i k \frac{[\lambda(i+k-1)T_s]^k}{k!} \exp[-\lambda(i+k-1)T_s]$$

$$(CL)_2 = (OL)_2$$

$$(CL)_1 = \sum_{i=1}^L i b_i$$

$$R_L = \frac{(OL)_1 - (CL)_1}{(OL)_1 + (OL)_2}$$

In the case of the output buffer, we define,

$$c_i = \text{Prob}[i \text{ items in the buffer, } B_2, \text{ at the end of the relaxation period}]$$

$$= \text{Prob}[(L-i) \text{ items leaving the buffer during the relaxation period}]$$

$$(OL)_1 = \mu NT_s$$

$$(CL)_1 = \sum_{i=0}^{L-1} (L-i) \cdot c_i$$

where μ is the mean arrival rate of retrieval requests.

R_L can be obtained simply by replacing λ by μ in the expressions derived for input buffer. For $\lambda = \mu$, $c_i = b_{L-i}$, and R_L will be identical for input and output buffers of equal length.

In the case of the heterogeneous process, λ_{\max} or μ_{\max} is chosen to compute R_L .

Analytical values for the fractional loss of information, R_L , in the buffered serial memory with constant clock ECM are recorded in Table 2.3 for $N = 512$, 1024 and 2048; $L = 2, 4, 6, 8, 12, 16, 20, 24, 28$ and 32; and $MRT/T_s = 25, 50, 75, 100, 150, 200, 250, 300$,

TABLE 2.3 Fractional loss of information (R_L) in buffered serial memory with constant clock ECM.
Analytical values.

N = 512										
MRT($\times T_s$) \longrightarrow										
L \	25	50	75	100	150	200	250	300	350	400
2	.8985	.8014	.7056	.6153	.4646	.3553	.2776	.2217	.1806	.1496
4	.7978	.6055	.5304	.2959	.1410	.0722	.0400	.0237	.0149	.0098
6	.6981	.4183	.2136	.1046	.0276	.0088	.0033	.0014	.00066	.00034
8	.5992	.2548	.0831	.0269	.0036	.00069	.00017	.00005	.00002	.00001
12	.4053	.0581	.0049	.00073	.00002	.00000				
16	.2276	.0064	.00017	.00001	.00000					
20	.0959	.00036	.00000							
24	.0280	.00001								
28	.0055	.00000								
32	.0007									
N = 1024										
2	.9491	.9005	.8518	.8031	.7070	.6166	.5355	.4654	.4060	.3569
4	.8985	.8014	.7042	.6081	.4321	.2970	.2036	.1414	.1002	.0724
6	.8481	.7027	.5579	.4208	.2148	.1051	.0527	.0277	.0153	.0088
8	.7979	.6045	.4156	.2568	.0836	.0270	.0094	.0036	.0015	.0007
12	.6981	.4106	.1745	.0587	.0059	.00073	.00011	.00002	.00000	
16	.5992	.2315	.0445	.0065	.00017	.00001	.00000			
20	.5011	.0978	.0064	.00036	.00000					
24	.4039	.0286	.00053	.00001	.00000					
28	.3081	.0056	.00072	.00000						
32	.2163	.00072	.00000							
N = 2048										
2	.9745	.9502	.9258	.9014	.8527	.8039	.7555	.7077	.6614	.6171
4	.9591	.9005	.8518	.8031	.7057	.6093	.5170	.4329	.3596	.2975
6	.9238	.8509	.7779	.7049	.5596	.4221	.3052	.2154	.1506	.1054
8	.8985	.8014	.7042	.6070	.4179	.2578	.1489	.0839	.0473	.0271
12	.8481	.7027	.5572	.4131	.1755	.0590	.0186	.0060	.0020	.00073
16	.7979	.6044	.4113	.2334	.0448	.0067	.00099	.00017	.00003	.00001
20	.7479	.8066	.2705	.0990	.0065	.00036	.00002	.00000		
24	.6981	.4091	.1485	.0290	.00053	.00001	.00000			
28	.6486	.3128	.0056	.00002	.00000					
32	.5992	.2200	.0203	.00073	.00000					

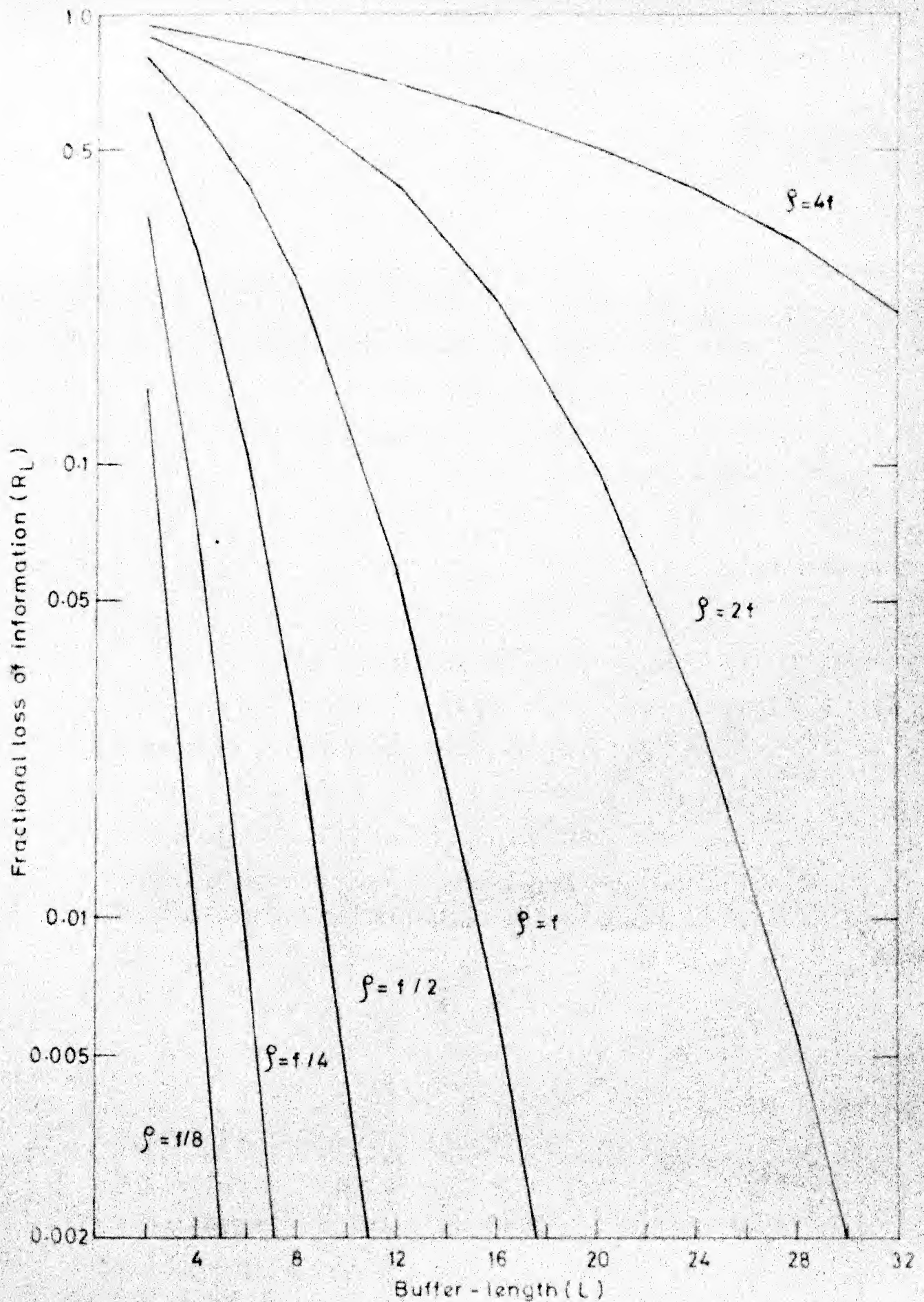


Figure 2.5 BUFFERED SERIAL MEMORY WITH CONSTANT CLOCK ECM, R_L VERSUS L .

350 and 400. To illustrate with an example, R_L for $L = 8$ and $MRT = 100 T_s$ can be obtained from this table as 0.02686 for $N = 512$ and 0.2568 for $N = 1024$. It is to be noted that R_L also depends on the product $\rho = \lambda NT_s$. The values for R_L are fairly close within an accuracy of 0.5 percent for a specified value of ρ obtained from different pairs of N and MRT . ρ can be expressed in terms of f (refer to Table 2.2). Figure 2.5 is a plot of R_L versus buffer-length for $\rho = 4f, 2f, f, \frac{1}{2}f, \frac{1}{4}f$ and $\frac{1}{8}f$.

2.8 VERIFICATION OF THE ABOVE MODEL

Discrete-event-simulation [G.S.Fishman, 1973] of the proposed scheme was carried out. A simulation program was written in GPSS-III and processed on IBM 7044. The whole system is subdivided into two practically independent subsystems: (1) Input buffer of length L , Poisson arrival, server-relaxation period of NT_s , and constant service time, T_s , (2) Output buffer of length L , constant inter-arrival time, T_s , source-relaxation period of NT_s and exponential service time. The first subsystem on the input end can be formulated as an M/G/1/L queueing model, i.e., Poisson arrival (M), General service time distribution (G), one server (1) and a queue of finite length (L). Input to the buffer is assumed to be from an infinite source of population. Another subsystem on the output end can be formulated as G/M/1/L queueing

model, i.e., general inter-arrival time (G), exponential service time (M), one server (1) and a queue of finite length (L). Data stored in SR is the source of input to the buffer. It is available to fill the buffer periodically. For practical purposes (i.e., for small values of L), it can be treated as an infinite source of population.

In the case of wait-system, an incipient request (transaction) examines the occupancy of buffer (storage entity). It gets immediate access if the buffer is not full, otherwise the request waits until the relaxation period is over. Waiting time for the earlier requests is zero. The time lapsed between the generation and the service instants is tabulated. Mean of the tabulated waiting time is the mean access time (T_b) which is used to compute the performance-improvement (η). Each simulation experiment runs for 5000 requests. A request is for insertion at the input end, whereas it is for retrieval at the output end. Hence MRT denotes mean inter-insertion time for M/G/1/L model and mean inter-retrieval time for G/M/1/L model.

The confidence-interval for the simulation results is computed as follows. For a fairly large sample of size S, the variance of the sample mean is estimated as σ^2/S and the distribution of the sample mean can be regarded as normal. Given the sample mean, m, and the sample-variance, σ^2 , the 95 percent confidence interval, C

for the population mean lies between $m - 1.96\sigma/\sqrt{S}$ and $m + 1.96\sigma/\sqrt{S}$. We obtain mean access time, T_b , and standard deviation, σ , from the simulation and compute η and its 95 percent confidence interval, $c = \frac{1.96\sigma}{T_u\sqrt{S}}$. The upper limit of a confidence interval should be interpreted such that η never exceeds 100 percent.

Tables 2.4(a) and (b) compare the analytical values of T_b and η for $L = 2, 4, 6, 8, 12$, and 16 ; $MRT/T_g = 25, 50, 75, 100, 150, 200, 300$ and 400 ; and $N = 512$ and 1024 .

Figure 2.6 is a plot of η versus L . Solid lines correspond to the simulation whereas dashed lines to the analytical results. The pair of curves marked with A corresponds to $MRT = 25T_g$. Similarly, those marked with B and C correspond to $MRT = 50 T_g$ and $100 T_g$ respectively. Simulation results agree well with analytical results.

The input buffer is formulated as an M/G/1/L model and the output buffer as a G/M/1/L model. Simulation study was carried out for both of these models. The values of R_L obtained from the simulation of these models are compared with those obtained analytically. These three values of R_L are recorded together as $(q, (s), [s])$ for $L = 2, 4, 8, 12, 16$ and 24 ; and $MRT/T_g = 100, 200, 300$ and 400 in Tables 2.5(a) for $N = 512$, (b) for $N = 1024$ and (c) for $N = 2048$. q corresponds to the analytical value. (s) and $[s]$ correspond to the values obtained from the simulation of M/G/1/L and G/M/1/L models

respectively. These values match within an accuracy of 0.5 percent. Hence, duality of these models is verified.

Table 2.6 records R_L for $L = 2, 4, 8, 12$ and 16 , and $N = 512$. The input process is considered to be a heterogeneous process whereas the output process a homogeneous process. MRT alternates between two levels, i.e., x and $3x$ for heterogeneous process. Each level extends until 600 retrieval-requests arrive. MRT remains fixed at $2x$ for homogeneous process. We consider $x = 50 T_s, 100 T_s, 150 T_s$ and $200 T_s$. Initial occupancy of SR is considered to be $N/5$. Each simulation experiment runs for 4800 requests.

When MRT is equal to x on the input end and to $2x$ on the output end, insertions are more than retrievals by roughly 300 hence $N > 300$. Because of the availability of SRs in sizes of 256, 512, 1024 etc., we consider $N = 512$. One can observe from Tables 2.3 and 2.6 that the fractional loss of insertion-requests with two-level MRTs, namely x and $3x$ is close to the average of R_L for $MRT = x$ and $3x$ respectively. To illustrate with an example, we consider $N = 512$ and $L = 4$. In this configuration, $R_L = 0.2959$ for $MRT = 100 T_s$ and $R_L = 0.0237$ for $MRT = 300 T_s$. R_L for the heterogeneous input process with $100 T_s$ and $300 T_s$ is tabulated as 0.1335 whereas average value is 0.1598. Occupancy of SR reaching the

TABLE 2.4(a) Performance-improvement (η) of buffered serial memory with constant clock ECM. Comparison of analytical results with the simulation results. $N = 512$.

SIMULATION						ANALYTICAL					
MRT ($\times T_s$)	L	SIMULATION				MRT ($\times T_s$)	L	ANALYTICAL			
		T_b	$\eta \pm c$	T_b	η			T_b	η	T_b	η
25	2	111.9	56.7 \pm 4.2	143.5	43.9	50	2	90.5	64.9 \pm 4.2	120.7	53.2
	4	65.3	74.7 \pm 3.8	76.1	70.3		4	44.9	82.9 \pm 3.9	52.3	79.6
	6	44.6	82.9 \pm 2.4	47.3	81.5		6	24.7	90.7 \pm 2.0	24.5	90.4
	8	31.9	87.9 \pm 1.4	21.3	87.8		8	16.5	93.8 \pm 1.5	10.8	95.8
	12	16.7	93.8 \pm 1.0	14.3	94.4		12	6.6	97.7 \pm 1.2	1.4	99.4
	16	8.9	96.9 \pm 0.9	5.8	97.7		16	4.7	98.5 \pm 1.0	0.1	100.0
75	2	77.0	70.0 \pm 3.8	96.3	62.4	100	2	68.9	73.5 \pm 4.0	75.7	70.4
	4	33.7	87.2 \pm 2.8	32.1	87.5		4	26.6	89.9 \pm 2.9	18.9	92.6
	6	14.9	94.6 \pm 2.0	10.1	96.0		6	9.9	96.5 \pm 2.2	4.1	98.4
	8	6.0	97.8 \pm 1.4	2.7	98.9		8	4.0	98.9 \pm 1.1	0.7	99.7
	12	3.0	99.3 \pm 0.1	0.1	100.0		12	1.1	99.6 \pm 0.4	0.0	100.0
	16	0.0	100 -0.1	0.0	100.0		16	0.0	100.0-0.1	0.0	100.0
150	2	52.6	89.7 \pm 3.2	46.4	81.9	200	2	37.6	85.6 \pm 2.4	29.4	88.5
	4	13.5	95.0 \pm 2.6	6.8	97.4		4	6.2	97.7 \pm 1.6	2.7	98.9
	6	2.7	99.3 \pm 1.2	0.8	99.7		6	0.8	99.9 \pm 0.7	0.2	99.9
	8	1.2	99.7 \pm 0.9	0.1	100.0		8	0.3	100.0-0.2	0.0	100.0
	12	0.7	99.9 \pm 0.5	0.0	100.0		12	0.0	100.0-0.1	0.0	100.0
	16	0.0	100 -0.1	0.0	100.0		16	0.0	100.0	0.0	100.0
300	2	26.8	89.9 \pm 3.7	13.5	94.7	400	2	14.3	94.4 \pm 3.2	7.2	97.2
	4	8.1	96.9 \pm 3.0	0.6	99.8		4	1.2	99.7 \pm 1.2	0.2	99.9
	6	1.0	99.7 \pm 0.7	0.0	100.0		6	0.5	100.0-0.2	0.0	100.0
	8	0.9	99.9 \pm 0.2	0.0	100.0		8	0.0	100.0	0.0	100.0
	12	0.2	100.-0.1	0.0	100.0		12	0.0	100.0	0.0	100.0
	16	0.0	100.0	0.0	100.0		16	0.0	100.0	0.0	100.0

TABLE 2.4(b) Performance-improvement (η) of buffered serial memory with constant clock BCM. Comparison of analytical results with the simulation results. $N = 1024$

SIMULATION						ANALYTICAL					
MRT ($\times T_s$)	L	T_b	$\eta \pm c$	T_b	η	MRT ($\times T_s$)	L	T_b	$\eta \pm c$	T_b	η
25	2	234.6	54.3 \pm 4.0	311.6	39.1	50	2	217.1	57.7 \pm 4.0	289.3	43.5
	4	144.4	71.9 \pm 2.8	176.7	65.5		4	131.2	74.5 \pm 3.2	153.5	70.0
	6	107.0	79.3 \pm 2.6	119.0	76.7		6	83.6	83.8 \pm 2.2	95.4	81.4
	8	78.1	84.8 \pm 1.8	87.1	83.0		8	60.8	88.3 \pm 1.3	63.0	87.7
	12	53.1	89.7 \pm 0.7	52.9	89.7		12	29.7	94.3 \pm 0.8	28.8	94.4
	16	35.4	93.2 \pm 0.4	34.7	93.2		16	11.7	97.7 \pm 0.4	11.7	97.7
75	2	202.2	60.6 \pm 3.6	265.1	48.2	100	2	186.2	63.7 \pm 3.2	240.6	53.0
	4	111.8	78.4 \pm 2.2	129.1	74.8		4	89.3	82.7 \pm 2.2	105.1	79.5
	6	66.5	87.2 \pm 1.9	71.2	86.1		6	48.4	90.7 \pm 1.4	49.3	90.4
	8	37.1	92.8 \pm 1.2	39.7	52.2		8	25.8	95.2 \pm 1.3	21.7	95.8
	12	14.7	97.3 \pm 0.9	10.4	98.0		12	5.3	99.1 \pm 1.0	2.9	99.4
	16	6.1	98.9 \pm 0.8	1.8	99.7		16	1.8	99.9 \pm 0.5	0.2	100.0
150	2	150.0	70.9 \pm 2.7	193.1	62.3	200	2	128.9	75.0 \pm 3.0	151.6	70.4
	4	60.6	88.3 \pm 2.2	64.4	87.4		4	42.5	91.8 \pm 2.1	37.8	92.6
	6	28.7	94.6 \pm 1.7	20.3	96.0		8	17.0	96.9 \pm 1.4	8.1	98.4
	8	10.5	98.1 \pm 1.0	5.4	98.9		8	2.3	99.6 \pm 0.4	1.4	99.7
	12	6.0	98.9 \pm 0.7	0.2	100.0		12	0.1	100.0 \pm 0.2	0.0	100.0
	16	4.7	99.3 \pm 0.5	0.0	100.0		16	0.0	100.0	0.0	100.0
300	2	96.6	81.3 \pm 2.2	92.8	81.9	400	2	78.1	84.8 \pm 2.6	98.8	88.5
	4	21.6	95.9 \pm 1.6	13.6	97.4		4	14.2	97.3 \pm 1.5	5.5	98.9
	6	4.5	99.3 \pm 0.9	1.5	99.7		6	1.6	99.9 \pm 0.2	0.4	99.9
	8	1.1	99.9 \pm 0.2	0.1	100.0		8	0.4	100.0 \pm 0.2	0.0	100.0
	12	0.1	100.0 \pm 0.1	0.0	100.0		12	0.0	100.0 \pm 0.1	0.0	100.0
	16	0.0	100.0	0.0	100.0		16	0.0	100.0	0.0	100.0

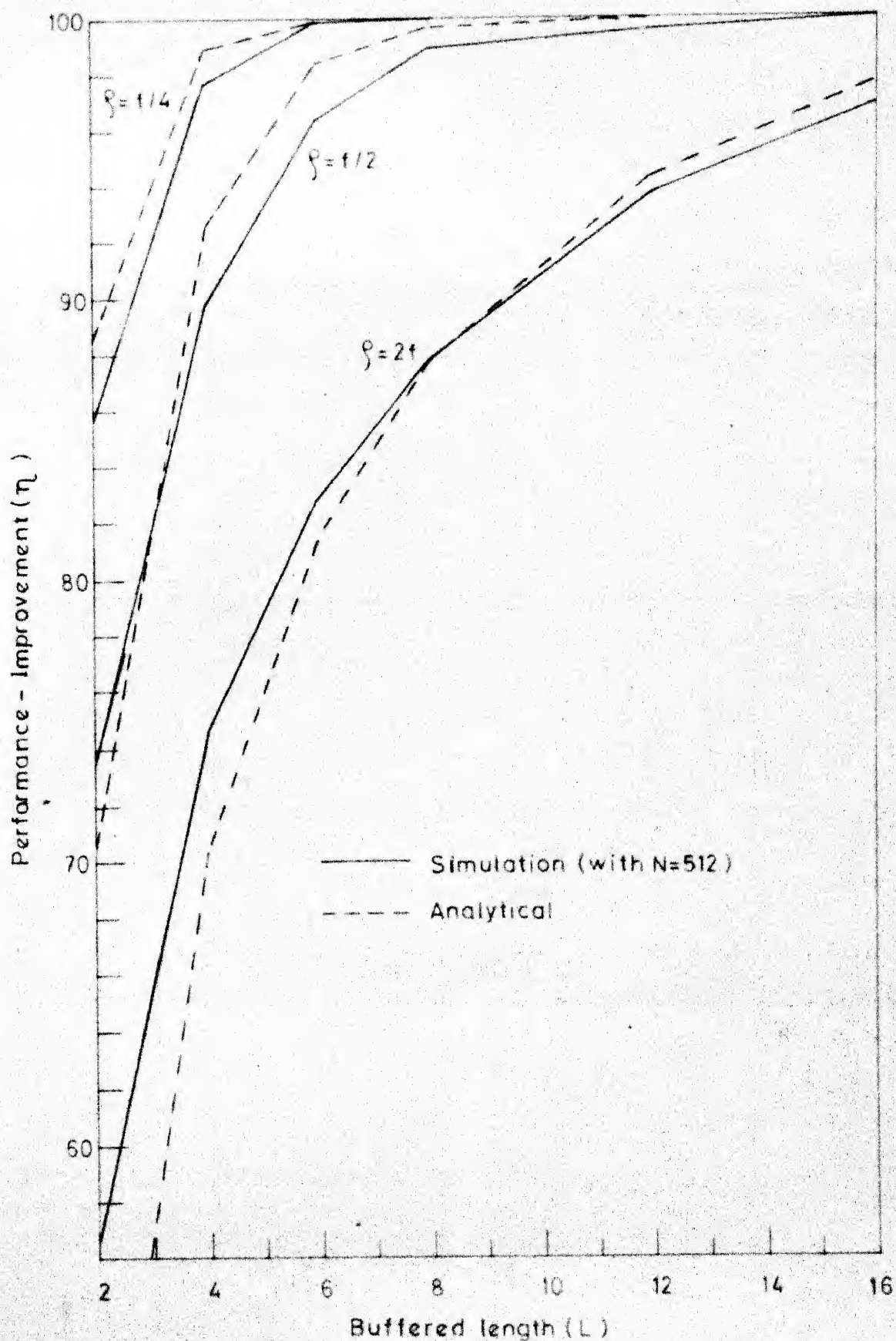


Figure 2.6 BUFFERED SERIAL WITH CONSTANT CLOCK ECT.

TABLE 2.5 Fractional loss of information.
Comparison of analytical values with simulation
values of M/G/l/L and G/M/l/L Models.

N = 512

MRT ($\times T_s$)	L =	2	4	8	12	16	24
100		.6153	.2959	.0269	.0007	.0000	
		(.6122)	(.2934)	(.0298)	(.0010)	(.0000)	
		[.6148]	[.3007]	[.0287]	[.0006]	[.0000]	
200		.3553	.0722	.0007	.0000		
		(.3493)	(.0706)	(.0008)	(.0000)		
		[.3593]	[.0730]	[.0005]	[.0000]		
300		.2217	.0237	.0000			
		(.2213)	(.0240)	(.0000)			
		[.2213]	[.0248]	[.0000]			
400		.1496	.0098	.0000			
		(.1298)	(.0080)	(.0000)			
		[.1493]	[.0098]	[.0000]			

TABLE 2.5 (Continued)

N = 1024

MRT (x T _s)	L = 2	4	8	12	16	24
100	.8031 (.8031) [.8046]	.6081 (.6050) [.6124]	.2568 (.2517) [.2696]	.0597 (.0576) [.0538]	.0065 (.0074) [.0081]	.0000 (.0000) [.0000]
200	.6165 (.6160) [.6142]	.2970 (.3006) [.3070]	.0270 (.0282) [.0290]	.0007 (.0007) [.0006]	.0000 (.0000) [.0000]	
300	.4654 (.4566) [.4624]	.1414 (.1396) [.1388]	.0036 (.0040) [.0033]	.0000 (.0000) [.0000]		
400	.3559 (.3680) [.3633]	.0724 (.0730) [.0755]	.0007 (.0010) [0.006]	.0000 (.0000) [.0000]		

N = 2048

100	.9014 (.8985) [.9016]	.8031 (.8010) [.8002]	.6070 (.6005) [.6122]	.4131 (.3905) [.4306]	.2334 (.2345) [.2378]	.0290 (.0280) [.0257]
200	.8039 (.8075) [.8024]	.6093 (.6105) [.6118]	.2578 (.2543) [.2666]	.0590 (.0644) [.0626]	.0066 (.0058) [.0074]	.0000 (.0000) [.0000]
300	.7077 (.7120) [.7056]	.4329 (.4315) [.4346]	.0840 (.0840) [.0838]	.0060 (.0066) [.0070]	.00017 (.0000) [.0003]	.0000 (.0000) [.0000]
400	.6171 (.6205) [.6141]	.2975 (.3093) [.3012]	.0271 (.0260) [.0225]	.0007 (.0006) [.0008]	.0000 (.0000) [.0000]	

Simulation results (.) corresponds to M/G/l/L model and
 [.] corresponds to G/M/l/L model.

TABLE 2.6 Heterogeneous input and homogeneous output processes.

N = 512. 600 retrieval-request per level.

MRT		CAPACITY		MEAN OCCUPANCY		SR OCCUPANCY		LOSS OF INFO.	
INS.	RET.	B ₁	B ₂	B ₁	B ₂	MEAN	MAX.	INS.	RET.
50	100	2	2	1.3	1.6	63	128	.7175	.6113
		4	4	1.9	1.6	71	192	.3478	.3459
		8	8	2.2	5.1	119	320	.1079	.0797
		12	12	2.5	9.4	175	384	.0283	.0034
		16	16	2.4	13.5	204	448	.0027	.0010
100	200	2	2	0.9	1.0	69	192	.3981	.3637
		4	4	1.1	2.6	102	320	.1335	.1138
		8	8	1.1	6.5	176	384	.0131	.0195
		12	12	1.2	10.9	257	512	.0004	.0002
150	300	2	2	0.6	1.2	63	256	.2635	.2531
		4	4	0.7	3.2	192	448	.0589	.0331
		8	8	0.8	7.2	205	384	.0004	.0008
		12	12	0.8	11.1	227	448	.0000	.0000
200	400	2	2	0.5	1.4	108	256	.1779	.1588
		4	4	0.5	3.5	217	448	.0277	.0068
		8	8	0.6	7.5	262	512	.0002	.0005

value of its capacity can also be observed from Table 2.6. For the buffers of equal length, the insertion-loss is more than the retrieval-loss. This behoves us to increase the size of the input buffer.

2.9 BUFFER UTILIZATION

In order to find the mean occupancy of buffer, we can employ Skinner's model [C.E. Skinner, 1967] with a constraint that the queue is of finite length. Skinner considered a problem where the server walks away for a period of time when the queue becomes empty. Customers join a queue which may be of any length. We develop a computationally convenient analytic method for the analysis of two classes of finite queueing systems, namely, $M/G/1/L$, i.e., Poisson arrivals (M), general service time (G), one server (1) and finite queue (L); and $G/M/1/L$, i.e. general arrival (G), exponential service time (M), one server (1) and finite queue (L).

A customer corresponds to a request. Similarly a queue corresponds to a buffer. Service in $M/G/1/L$ corresponds to transfer from B_1 to SR. Arrival in $G/M/1/L$ corresponds to transfer from SR to B_2 , and service time corresponds to inter-arrival time for retrieval-requests.

2.9.1 $M/G/1/L$ WITH SERVER-RELAXATION TIME

B_1 is emptied into SR during consecutive shift-periods. The tail of the information-chunk in SR leaves

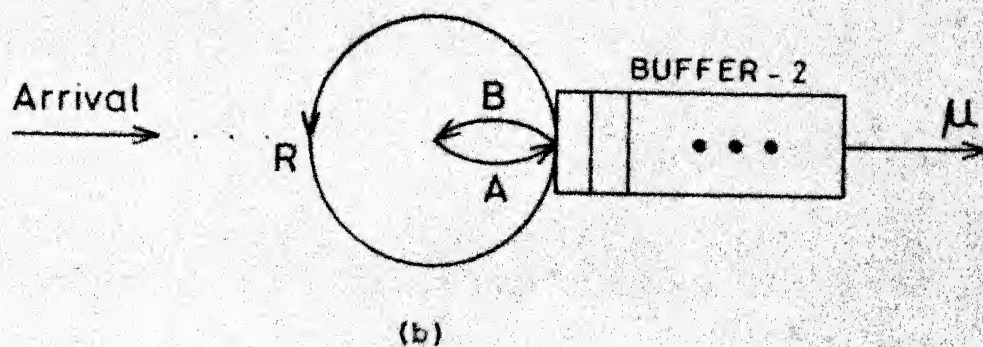
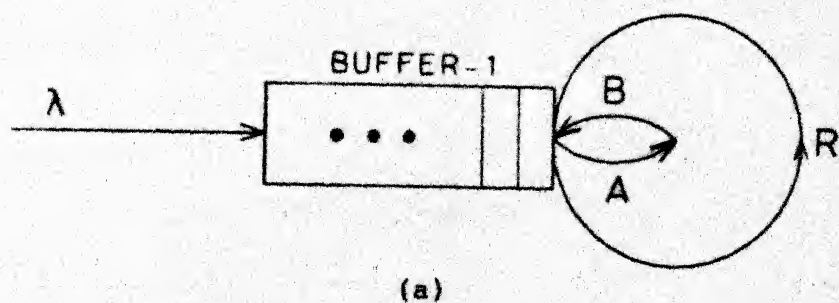


Figure 2.7 FINITE QUEUE SKINNER'S MODELS FOR
 (a) M/G/1/L WITH SERVER-RELAXATION TIME
 (b) G/M/1/L WITH SOURCE-RELAXATION TIME

the input port and inspects B_1 again after the relaxation period, NT_s . Figure 2.7(a) depicts a finite queue Skinner's model [cf. C.E. Skinner, 1967]. In our example, relaxation period $R = NT_s$, service time $A = T_s$ and latency period $B = 0$. Requests for insertion are rejected if B_1 is found full. Insertion requests follow Poisson distribution with mean arrival rate of λ . We define,

$$g_i = \text{Prob}[i \text{ information bits in } B_1 \text{ at the beginning of transfer from } B_1 \text{ to SR}]$$

and

$$h_i = \text{Prob}[i \text{ information bits left in } B_1 \text{ at the instant of a transfer-completion}]$$

$$\text{Let } \rho_N = \lambda NT_s, \quad \rho_1 = \lambda T_s$$

$$g_i = g_0 e^{-\rho_N} \frac{\rho_N^i}{i!} + \sum_{k=1}^{i+1} g_k e^{-\rho_1} \frac{\rho_1^{i+1-k}}{(i+1-k)!}$$

$$\text{for } i = 0, 1, \dots, (L-1)$$

and

$$g_L = g_0 \sum_{j=L}^{\infty} e^{-\rho_N} \frac{\rho_N^j}{j!} + \sum_{k=1}^L g_k \sum_{j=L-k+1}^{\infty} e^{-\rho_1} \frac{\rho_1^j}{j!}$$

The last equation is linearly dependent on the previous ones and hence can be replaced by the probability conservation constraint $\sum_{i=0}^L g_i = 1$. The resulting set of equations can be solved by iterative method.

$$\begin{bmatrix} a_{00} & a_{01} & 0 & \dots & 0 \\ a_{10} & a_{11} & a_{12} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{L-1,0} & a_{L-1,1} & a_{L-1,2} & \dots & a_{L-1,L} \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{L-1} \\ g_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

After computing g_0, g_1, \dots, g_L , we can compute h_i as follows:

$$h_i = \sum_{k=1}^{i+1} \frac{g_k}{1-g_0} e^{-\rho_1} \frac{\rho_1^{i+1-k}}{(i+1-k)!}$$

for $i = 0, 1, \dots, (L-2)$

$$\begin{aligned}
 h_{L-1} &= \sum_{k=1}^L \frac{g_k}{1-g_0} \sum_{j=L-k}^{\infty} e^{-\rho_1} \frac{\rho_1^j}{j!} \\
 &= (1 - \sum_{i=0}^{L-2} h_i)
 \end{aligned}$$

Since insertion-request and transfer operation are independent, h_i also denotes the probability of i -information bits in B_1 at the instant of arrival of an insertion-request. Thus, we can compute the mean buffer occupancy, \bar{B}_1 , as follows.

$$\bar{B}_1 = \sum_{i=0}^{L-1} (i+1)h_i.$$

2.9.2 G/M/1/L WITH SOURCE-RELAXATION TIME

If we reverse the direction of flow of requests and transfer of data, the resulting queuing system (Figure 2.7(b)) is G/M/1/L, i.e., general inter-arrival times (G), exponential service times (M), one server (1), and finite queue (L). In our example, information-chunk (ic) is the source of population, its head inspects buffer (queue). If the buffer is not full, service-inspection cycle begins. Transfer from SR into B_2 corresponds to service. Transfer takes place during consecutive shift-periods. If the buffer is found full, ic begins an inactivity period of R which is the same as the relaxation period, NT_s . Retrieval requests follow Poisson distribution with mean arrival rate of μ .

We define,

$$g'_i = \text{Prob}[i \text{ information bits in } B_2 \text{ at the beginning of the transfer from SR into } B_2]$$

and

$$h'_i = \text{Prob}[i \text{ information bits in } B_2 \text{ at the instant of a transfer-completion}]$$

Refer to Figure 2.7(b), relaxation period $R = NT_s$, inter-arrival time, $A = T_s$ and latency time, $B = 0$. Let $\sigma_N = \mu NT_s$ and $\sigma_1 = \mu T_s$. We can derive the occupancy probabilities from the following set of equations,

$$g'_0 = g'_L \sum_{j=L}^{\infty} \frac{\sigma_N^j}{j!} e^{-\sigma_N} + \sum_{k=0}^{L-1} g'_k \sum_{j=k+1}^{\infty} \frac{\sigma_1^j}{j!} e^{-\sigma_1}$$

$$g'_i = g'_L \frac{\sigma_N^{L-i}}{(L-i)!} e^{-\sigma_N} + \sum_{k=i-1}^{L-1} g'_k \frac{\sigma_1^{k+1-i}}{(k+1-i)!} e^{-\sigma_1}$$

for $i = 1, 2, \dots, L$

The first equation is dependent upon the rest of the equations. Hence it can be replaced by the probability conservation constraint, that is $\sum_{i=0}^L g'_i = 1$.

$$\begin{bmatrix} 1 & 1 & \dots & 1 & 1 \\ a'_{10} & a'_{11} & \dots & a'_{1(L-1)} & a'_{1L} \\ 0 & a'_{21} & \dots & a'_{2(L-1)} & a'_{2L} \\ \vdots & & & \vdots & \vdots \\ 0 & 0 & \dots & a'_{L(L-1)} & a'_{L,L} \end{bmatrix} \begin{bmatrix} g'_0 \\ g'_1 \\ g'_2 \\ \vdots \\ g'_L \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

For $\sigma_1 = \rho_1$ and $\sigma_N = \rho_N$, we obtain $g'_i = g_{L-i}$ and $a'_{ij} = a_{(L-i)j}$.

After rewriting the above set of equations, we obtain a new set of equations identical to that in the case of M/G/1/L. This holds for all values of L. Thus we conclude that M/G/1/L queueing system with server-relaxation time is dual of G/M/1/L queueing system with source-relaxation time.

TABLE 2.7 M/G/1/L Finite queue Skinner's Model, N = 512.

S : Simulation result

A : Analytical result

(a) MRT = 100 T_s

L	\bar{q}	i =	0	1	2	3	4	5	6	7	8
2	S { 1.503	g_i	{ .334	.335	.331						
	A { 1.497		{ .336	.337	.327						
	S {	h_i	{ .503	.497							
	A {		{ .503	.497							
4	S { 2.377	g_i	{ .219	.218	.212	.191	.160				
	A { 2.396		{ .216	.217	.210	.193	.164				
	S {	h_i	{ .279	.271	.245	.205					
	A {		{ .274	.268	.246	.211					
8	S { 3.390	g_i	{ .166	.167	.161	.148	.126	.096	.068	.041	.026
	A { 3.397		{ .166	.166	.161	.148	.126	.097	.068	.043	.025
	S {	h_i	{ .200	.193	.177	.151	.115	.082	.050	.031	
	A {		{ .197	.193	.178	.151	.117	.082	.052	.030	
			0	2	4	6	8	10	12	14	16
12	S { 3.598	g_i	{ .162	.160	.120	.070	.020	.006	.0009		
	A { 3.563		{ .162	.158	.123	.066	.024	.006	.001		
	S {	h_i	{ .194	.171	.111	.051	.016	.003			
	A {		{ .192	.173	.114	.050	.015	.003			
16	S { 3.563	g_i	{ .162	.157	.122	.067	.024	.006	.0009	.0000	.000
	A { 3.570		{ .162	.158	.123	.066	.024	.006	.0010	.0001	.000
	S {	h_i	{ .194	.173	.114	.051	.015	.003	.0004	.0000	
	A {		{ .192	.173	.114	.050	.015	.003	.0005	.0000	

MIT = Mean Interarrival time

L = Maximum length of the queue

 \bar{q} = Mean queue length.

TABLE 2.7 (Continued)

(c) MRT = 300 T_s

S : Simulation result
A : Analytical result

L	\bar{q}	i =	0	1	2	3	4	5	6	7	8
S	1.381	g_i	.434	.351	.215						
A	1.385		.430	.352	.219						
S		h_i	.620	.379							
A			.615	.385							
S	1.771	g_i	.372	.307	.191	.092	.035				
A	1.771		.374	.307	.191	.092	.035				
S		h_i	.489	.309	.146	.056					
A			.490	.306	.147	.057					
S	1.817	g_i	.375	.303	.190	.088	.033	.009	.002	.000	.000
A	1.856		.369	.303	.188	.090	.035	.011	.003	.000	.000
S		h_i	.485	.304	.140	.053	.014	.003	.000	.000	
A			.478	.299	.144	.055	.018	.005	.001	.000	
			0	2	4	6	8	10	12	14	16
S	1.855	g_i	.372	.186	.035	.002	.000	.000	.000		
A	1.857		.367	.188	.035	.003	.0001	.000	.000		
S		h_i	.480	.143	.019	.001	.000	.000			
A			.478	.144	.018	.001	.000	.000			
S	1.872	g_i	.368	.187	.037	.004	.000	.000	.000	.000	.000
A	1.857		.369	.188	.035	.003	.0001	.000	.000	.000	.000
S		h_i	.478	.142	.019	.001	.000	.000	.000	.000	
A			.478	.144	.018	.001	.000	.000	.000	.000	

MIT = Mean interarrival time

L = Maximum length of the queue

 \bar{q} = Mean queue length

TABLE 2.7 (Continued)

S : Simulation result
A : Analytical result

(d) MRT = 400 T_s

L	\bar{q}	i =	0	1	2	3	4	5	6	7	8
2	S { 1.347	g_i	{ .474	.344	.182						
A { 1.338											
S	h_i	{ .653	.347								
A											
4	S { 1.588	g_i	{ .445	.321	.160	.058	.017				
A { 1.607											
S	h_i	{ .557	.288	.105	.030						
A											
8	S { 1.644	g_i	{ .437	.317	.161	.060	.021	.004	.000	.000	.000
A { 1.641											
S	h_i	{ .563	.285	.107	.037	.007	.001	.000	.000	.000	.000
A											
			0	2	4	6	8	10	12	14	16
2	S { 1.667	g_i	{ .433	.164	.021	.001	.000	.000	.000		
A { 1.642											
S	h_i	{ .551	.113	.008	.000	.00	.000				
A											
16	S { 1.630	g_i	{ .436	.160	.016	.001	.000	.000	.000	.000	.000
A { 1.642											
S	h_i	{ .568	.108	.007	.0002	.000	.000	.000	.000	.000	.000
A											

MIT = Mean interarrival time

L = Maximum length of the queue

 \bar{q} = Mean queue length.

2.9.3 VERIFICATION OF THE FINITE QUEUE SKINNER'S MODEL

In order to verify the analytical results of the M/G/1/L finite queue Skinner's model, the simulation study of the same is carried out. Each simulation experiment runs for 5000 requests. Mean buffer occupancy, \bar{q} ; and occupancy probabilities, g_i and h_i , are recorded in Tables 2.7(a-d) for $N = 512$; $L = 2, 4, 8, 12$ and 16 ; $MRT/T_s = 100, 200, 300$ and 400 .

It is observed that the analytical results agree well with those obtained from the simulation.

2.10 DISCUSSION OF RESULTS

We consider two measures of performance, namely, mean access time and fractional loss of information. Table 2.1 serves as a design-guide table for the improvement in performance (in terms of mean access time) as the design-criterion. For the fractional loss of information, Table 2.3 is to be referred to. Tables 2.4 and 2.5 compare the analytical results with the simulation results for performance-improvement (η) and fractional loss of information (R_L) respectively.

For R_L , the analytical values agree well with the simulation values. Figure 2.5 shows an exponential decay in R_L with an increase in buffer-length. In order to achieve $R_L \leq 10^{-3}$, one may select $L = 6, 8, 12, 20$ and 32 for $\rho = \frac{1}{8}f, \frac{1}{4}f, \frac{1}{2}f, f$ and $2f$ respectively. (N, MRT) pairs of $(512, 50 T_s)$; $(1024, 100 T_s)$; $(2048, 200 T_s)$ and $(4096, 400 T_s)$ give rise to $\rho = NT_s/MRT = 10.24 = f$.

For η , the analytical values are close to the simulation values. For L much greater than $\frac{1}{2}f$, analytical value of η is slightly more than the corresponding simulation value. For L much smaller than $\frac{1}{2}f$, analytical value of η is less than the corresponding simulation value. Figure 2.6 shows rapid increase in η with small increase in buffer-length for L less than ρ . To illustrate with an example, we consider $N = 512$ and $MRT=100T_g$ to provide $\rho = 5.12$. An addition of a buffer of length 2 exhibits about 70 percent improvement in performance. In order to achieve $\eta \geq 95$ percent, one may select $L = 2, 4, 6, 8$ and 16 for $\rho = \frac{1}{8}f, \frac{1}{4}f, \frac{1}{2}f, f$ and $2f$, respectively.

Table 2.6 records R_L for heterogeneous input and homogeneous output processes. Fractional loss of insertion requests with alternating MRT can be approximated by the average of R_L for respective MRT. Retrieval loss is slightly more than the expected value from the Table 2.3. This results because the assumption of infinite source of population for G/M/1/L model does not hold good when SR becomes empty.

Finite queue Skinner's models for M/G/1/L and G/M/1/L are simulated. The analytical results agree well with the simulation results.

2.11 APPLICATIONS

The proposed buffered serial memory is useful with heterogeneous input and output processes [G.Philokyprou, 1974]. Typical application areas include data compression, data acquisition systems, peripheral-CPU interfacing, time-shared systems, and communication networks with store and forward facility [R.C. Chen, et.al., 1976].

2.12 CONCLUSION

A buffered serial memory with constant clock ECMs is proposed. The entire system can be studied by analysing two practically independent subsystems: (1) input process, input buffer and periodic emptying of buffer, and (2) output process, output buffer and periodic filling of buffer. The first subsystem is formulated as an M/G/1/L queueing system with server-relaxation time, that is, a finite queue Skinner's model; and the second as a G/M/1/L queueing system with source-relaxation time. Both of these models are duals. A computationally convenient analytic method is developed to compute buffer-occupancy probabilities and mean buffer-length. The improvement in performance (η) over an unbuffered serial memory, and the fractional loss of information (R_L) are considered the design criteria to obtain an optimal length of buffer. Analytical values of η and R_L agree well with the simulation results.

CHAPTER 3

BUFFERED STACK MEMORY WITH VARIABLE CLOCK BCM

Stacks are widely used in various fields, namely, multiprocessing, dynamic programming, evaluation of recursive functions etc. Major shortcoming of a stack-based machine is a degradation in array processing. A buffered stack overcomes this shortcoming to a large extent. A scheme for a buffered stack organization is proposed. This consists of a large size unidirectional (static) shift-register stack (SS) and a small size bidirectional fast (buffer) stack (FS). Performance-improvement of a buffered stack over an unbuffered stack consisting of entirely unidirectional SRs is considered the design criterion to determine the size of buffer (FS).

3.1 INTRODUCTION

A stack is a storage unit with Last-In-First-Out (LIFO) access discipline. Stacks can be used in various ways, namely, providing a 'one for one' correspondence between source language operators and the machine instructions [R.J.Evey, 1963]; preserving code and data conditions to facilitate rapid environment change and hence saving an overhead for unlimited nesting of procedures; providing re-entrant capability of code to facilitate recursion; allowing fast interrupt handling;

increasing the speed of computation by virtue of code compression which is inherited from its implicit addressing [E.I.Organick, 1973; Hauck and Dent, 1968; C.B. Carlon, 1963]. Application of stacks can be noticed in multiprocessing subroutine operations, language translation, dynamic programming problems, efficient processing of object program written in a structured language like ALGOL, and facilitating GOTO-less programming by maintaining a control environment [Evey, 1963; Burns and Savilt, 1973; Bigelow, 1975].

Stack-based machines, like B-6700 suffer in the area of array processing where a great deal of indexing is involved. Barton [1970] suggests an adequately large 'fast stack top', say 32 fast registers on a slow stack, in order to solve the problem of great deal of indexing encountered in array processing. Enlarged fast stack top further offers a convenient opportunity to introduce hardware supported vector operations.

A stack can be simulated by software. However, a hardware stack offers a substantial increase in the speed of computation.

A stack can be formed by using a set of unidirectional shift-registers with some logic circuitry associated with it. Such a stack forms an asymmetric memory [J.G. Williams, 1973] as the ratio of insertion and

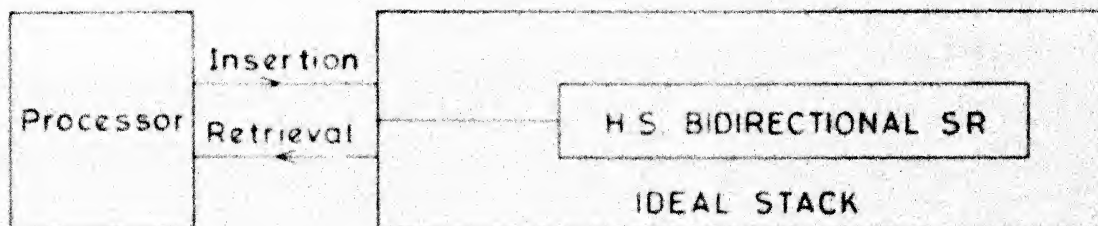
retrieval times is sufficiently large, i.e., equal to the size of stack minus one. A typical stack of size $(N+1)$ would require N circular shifts for an insertion in order to make the recently inserted word available on top of the stack, whereas only one shift is required for a consecutive retrieval. Mean access time will be $(N+1)/2$.

In this thesis a fast stack which consists of high speed registers or bidirectional shift-registers is suggested to be added to the unidirectional shift-register stack. This will improve the access time by virtue of enabling large number of requests to be satisfied from the fast stack.

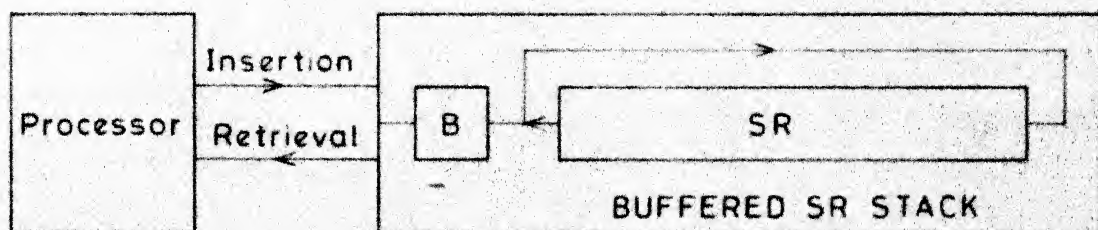
An asymmetric stack memory consisting of an SR of size 1K operating at 5 MHz exhibits the mean access time of 0.1 msec., that is 512 shift-periods. An addition of a small size fast stack will reduce it to a few shift-periods.

Processor accesses the stack by generating two kinds of requests (see Figure 3.1): (1) request for insertion, and (2) request for retrieval. These requests occur randomly and may be assumed to follow a particular probabilistic distribution.

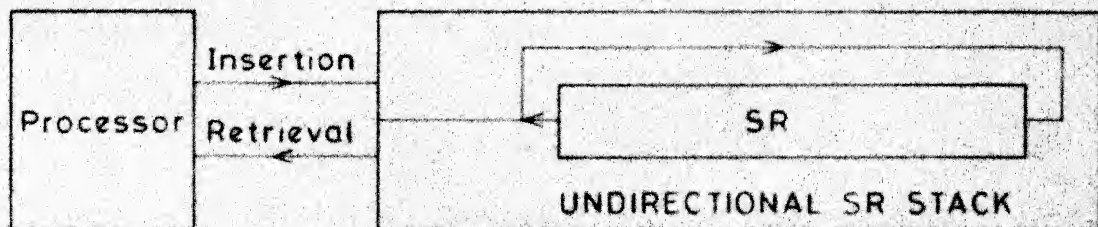
Our design objective is to develop a scheme so as to make the large size asymmetric stack look like an entire fast stack [Figure 3.1(a)] in performance.



(a)



(b)



(c)

Figure 3.1. STACK MEMORY ORGANIZATIONS.



Figure 3.2 BUFFERED STACK MEMORY WITH VARIABLE CLOCK BCM

In the following sections a scheme for a buffered stack memory is proposed and its mathematical model is developed to determine the optimal size of the fast stack.

3.2 DESCRIPTION OF THE SCHEME

Processor issues requests for insertion into and retrieval out of the stack memory (see Figure 3.1). We propose a buffered stack memory organization (Figure 3.2) consisting of large size ECMS with variable clock-rate (static SRs), a small size fast-stack (FS) and the associated circuitry. m N -bit shift-registers in parallel can store N -words, m -bit/word. These together with S -register form an asymmetric slow stack (SS). m L -bit bidirectional high speed shift-registers form a fast stack. Typically, N is in the range of 512 to 2048 and L is 8 to 32. Transfer-rate, $1/T_s$, of a shift-register may be 1-5 Mb/S. Processor accesses a data word via a buffer-register. Data word is transferred from the buffer register into the fast stack for insertion. Insertion request is assumed to be rejected at the instant the fast stack is found full and the processor has to wait until a room is created on the top of fast stack. Retrieval request enables the transfer of the top of fast stack into the buffer register. If FS is found empty the Processor has to wait until the head of the information-chunk (ic) reaches the output port. FS counter (f.s.c.) indicates the current content of FS. FLAG is a flip-flop which is set when a data word is

stored in both FS and SS. FLAG is reset when initially stored duplicate data word is shifted out due to a request for insertion. W_1 and W_2 are the wired OR junctions. We have discussed this scheme in an earlier paper [V.Rajaraman and Om Vikas, 1975].

3.3 OPERATION

Whenever there is any request for insertion or retrieval, FS is inspected and depending on its occupancy it is either satisfied immediately or required to wait until SS is ready for the transfer of a data word. As a request for insertion arrives, FS may be found in one of three possible states: empty, full and partially filled. If FS is found empty, the desired data word is written into FS and the S-register. Subsequently FLAG is set. FS is made available immediately for a next request, whereas the content of SS will be circularly shifted N times to make the recent insertion available on the top of the stack. SS is assumed of size $(N+1)$. There will be overlapping of the operations on FS with a write operation into SS. If FS is found full, the flag is examined. If flag is found set which implies that the bottom of FS is already available as the top of SS, then FS is shifted down and no insertion into SS is initiated. At the same time the flag is reset. If the flag is found reset and FS is full, FS is shifted down into SS provided SS is available otherwise the system waits. Subsequently a

room on the top of FS is created for a new insertion. If FS is partially full, desired data word is inserted, immediately, into FS.

When a request for retrieval arrives, the top of FS is transferred to the buffer register provided the FS is non-empty. If FS is found empty, the flag is examined. If the flag is set which implies a duplicate data word in SS, the top of SS is shifted out and the flag is reset. Subsequently, the top of SS is transferred into the buffer-register.

3.4 CRITERIA OF GOODNESS

The main objective to develop the buffered SR stack organization is to achieve the performance which approaches that of the buffer and the price/bit which approaches that of SR.

Mean access time can be considered a measure of performance. The size of fast stack should be as small as possible in order to reduce the price/bit. Another consideration may be that the processor waiting time should not exceed its allowable waiting time.

3.4.1 PERFORMANCE-IMPROVEMENT

An ideal stack (Figure 3.1(a)) and an asymmetric unbuffered stack (Figure 3.1(c)) lie on the extreme ends of the spectrum of the stack memory organization. The

performance of a stack memory organization can be expressed with respect to that of the stack memory consisting entirely of the unidirectional shift-registers. We define the performance-improvement (η) of a buffered stack (Figure 3.1(b)) over an unbuffered stack (Figure 3.1(c)) as follows.

$$\eta = \left(1 - \frac{T_b}{T_u}\right) \times 100 \text{ percent}$$

where T_b = mean access time of the buffered stack, and
 T_u = mean access time of the unbuffered stack.

In the case of ideal stack, $T_b = 0$ and hence $\eta = 100$ percent. For an unbuffered stack, $T_b = T_u$ and hence $\eta = 0$ percent, that is no improvement.

In the following section, a mathematical model is developed to compute the mean access time, T_b .

3.5 MATHEMATICAL MODEL

3.5.1 DESCRIPTION

We use the following notations

- (N+1) : size of SS,
- $1/T_s$: maximum transfer rate of SS,
- L : size of FS,
- $1/T_f$: maximum transfer rate of FS,

T_i : time for an insertion into SS,
 T_r : time for a retrieval out of SS,
 T_m : minimum time between consecutive requests.

It is to be noted that $T_i = N T_s$ and $T_r = T_s$.
 $T_i \gg T_r$.

A stack is a memory with LIFO access discipline. The buffered stack memory organization can be formulated as a queueing model with LIFO service discipline. D.M.G. Wishart [1960] studied M/G/1 queueing system with LIFO service discipline and compared the results with those in the case of FIFO service discipline. He concludes that the busy period distribution, the mean queue length and the mean waiting time remain the same for both FIFO and LIFO service disciplines. However, variance of waiting time is less in FIFO than in LIFO, and the probability of congestion is more in FIFO than in LIFO. A customer is congested if he arrives to find the service point occupied and has to wait more than the unexpended service time of the customer receiving service.

We are concerned with the mean waiting time, hence the above model is equivalent to one with FIFO access discipline.

The simplest request pattern to handle mathematically is the Poisson process. This arrival process has been widely studied [W. Feller, 1968]. Poisson process

implies that the probability of k arrivals with an arrival-rate of γ in an arbitrary interval of time, t , is

$$\text{Prob}[k \text{ arrivals in interval } t] = \frac{(\gamma t)^k}{k!} e^{-\gamma t}$$

and the inter-arrival intervals have the exponential density function

$$\text{Prob}[\text{inter-arrival time} = t] = \gamma e^{-\gamma t}$$

Let λ and μ be the mean arrival rates of the insertion-and retrieval-requests separately. The inter-arrival intervals for requests of both kinds follow exponential density function, that is $(\lambda+\mu) e^{-(\lambda+\mu)t}$, and the probability of k arrivals of requests of both kinds in an arbitrary interval of time, t , is

$$\frac{[(\lambda+\mu)t]^k}{k!} e^{-(\lambda+\mu)t}.$$

3.5.2 ANALYSIS

The proposed stack memory organization (see Figure 3.2) can be viewed as M/M/1 queuing system, i.e., Poisson arrival of insertion requests (M), Exponential inter-retrieval time (M), and single server (1). State-transition-rate diagram is shown in Figure 3.3. The departure from the classic M/M/1 model is that this model allows non-nearest-neighbour transitions as well. Notion of flow conservation applies to any Markov Chain. Thus we may construct 'non-nearest-neighbour' system (Figure 3.3) and still apply flow conservation technique [L.Kleinrock, 1975].

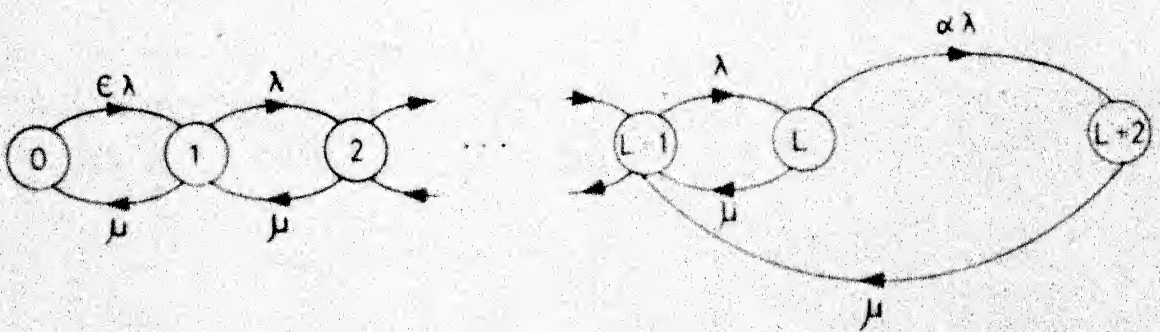


Figure 3.3 STATE-TRANSITION-RATE DIAGRAM

Let S_i denote the state with i items in the system. Notice that S_{L+1} is a transient state which results when FS is found full the first time after being empty. This state corresponds to shifting out the duplicate data word stored in FS. S_{L+1} is, virtually, S_L hence it is not shown in Figure 3.3. S_{L+2} results on the arrival of insertion request after shifting out the duplicate data word. $T_m = T_f < 1/(\lambda + \mu)$ leads to an assumption of instant response from FS for both insertions and retrievals. $T_r < 1/\mu$ also validates the assumption of instant response from SS for retrievals only. T_i may be referred to as the reorientation period. T_i^k denotes the reorientation period initiated in the state S_k .

Imminent insertion requests for the transition $S_0 \rightarrow S_1$ during T_i^0 and T_i^{L+2} are blocked. This is accounted for by introducing a factor $\epsilon \leq 1$. By the transition, $S_L \rightarrow S_{L+1}$, we mean shifting out the duplicate data word stored in FS. α accounts for the blocking of the transition $S_L \rightarrow S_{L+2}$ during T_i^0 and T_i^{L+2} .

Let $P_i(t)$ denote the probability that the system is in the state S_i at time t . The differential equations describing the proposed model are

$$P_0(t + \delta t) = \mu \delta t P_1(t) + (1 - \epsilon \lambda \delta t) P_0(t)$$

$$P_1(t + \delta t) = \mu \delta t P_2(t) + \epsilon \lambda \delta t P_0(t) + (1 - \lambda \delta t)(1 - \mu \delta t) P_1(t)$$

$$P_n(t+\delta t) = \mu \delta t P_{n+1}(t) + \lambda \delta t P_{n-1}(t) \\ + (1 - \lambda \delta t)(1 - \mu \delta t) P_n(t)$$

$$\text{for } 2 \leq n \leq L-2$$

$$P_{L-1}(t+\delta t) = \mu \delta t P_L(t) + \mu \delta t P_{L+2}(t) + \lambda \delta t P_{L-2}(t) \\ + (1 - \lambda \delta t)(1 - \mu \delta t) P_{L-1}(t)$$

$$P_L(t+\delta t) = \lambda \delta t P_{L-1}(t) + (1 - \alpha \lambda \delta t)(1 - \mu \delta t) P_L(t)$$

$$P_{L+2}(t+\delta t) = \alpha \lambda \delta t P_L(t) + (1 - \mu \delta t) P_{L+2}$$

We are primarily interested in the steady state solution. Consequently, the set of differential equations reduces to the following set of equilibrium (balance) equations.

$$\epsilon \lambda P_0 = \mu P_1$$

$$(\lambda + \mu) P_1 = \mu P_2 + \epsilon \lambda P_0$$

$$(\lambda + \mu) P_n = \mu P_{n+1} + \lambda P_{n-1}, \quad \text{for } 2 \leq n \leq L-2$$

$$(\lambda + \mu) P_{L-1} = \mu P_L + \mu P_{L+2} + \lambda P_{L-2}$$

$$(\alpha \lambda + \mu) P_L = \lambda P_{L-1}$$

$$\mu P_{L+2} = \alpha \lambda P_L$$

For $\beta = \lambda / \mu$,

$$P_1 = \epsilon \beta P_0,$$

$$P_n = \beta P_{n-1}, \quad 2 \leq n \leq L-1$$

$$P_L = \frac{\beta}{1 + \alpha \beta} P_{L-1}$$

$$P_{L+2} = \alpha \beta P_L$$

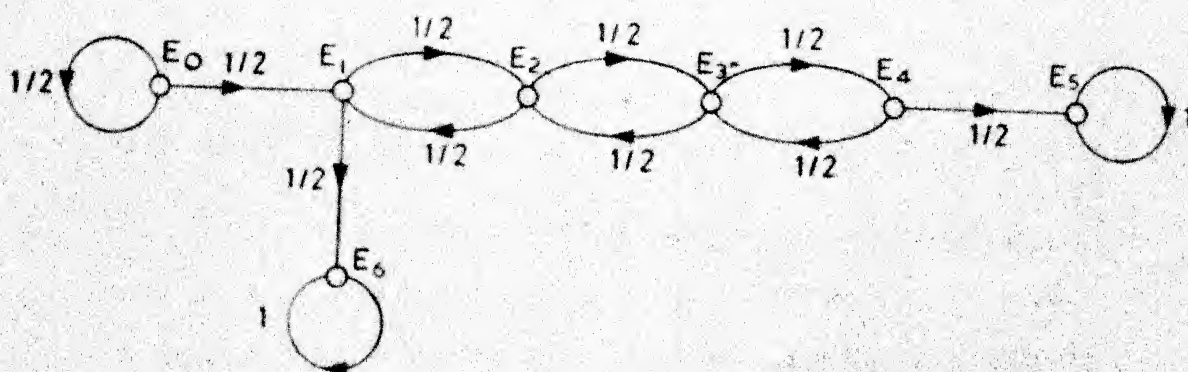


Figure 3.4 STATE-TRANSITION DIAGRAM

The following recurrence relation can be used to compute the probability vectors.

$$\pi(k+1) = \pi(k) \cdot M$$

where $\pi(k)$ is the probability vector at the k -th instant. Initial probability vector can be chosen as

$$\pi(1) = (1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0) .$$

Subsequently,

$$\pi(2) = (1/2 \ 1/2 \ 0 \ 0 \ 0 \ 0 \ 0)$$

$$\pi(3) = (1/4 \ 1/4 \ 1/4 \ 0 \ 0 \ 0 \ 1/4)$$

$$\pi(4) = (1/8 \ 1/4 \ 1/8 \ 1/8 \ 0 \ 0 \ 3/8)$$

$$\pi(5) = (1/16 \ 1/8 \ 3/16 \ 1/16 \ 1/16 \ 0 \ 1/2)$$

$$\pi(6) = (1/32 \ 1/8 \ 3/32 \ 1/8 \ 1/32 \ 1/32 \ 9/16)$$

and so on.

$$\pi_5(k) = \text{Prob}[\text{FS is full and SS is requested}]$$

$$\pi_6(k) = \text{Prob}[\text{FS becomes empty}]$$

$\pi_5(k)$ and $\pi_6(k)$ are the cumulative probabilities upto k steps. In the above example $\pi_5(6) = 1/32$, $\pi_6(6) = 9/16$.

Let $\varphi_1(j) = \text{Prob}[\text{first passage through } S_{L+1} \text{ at } j\text{-th instant}]$

$$\varphi_2(j) = \text{Prob}[\text{first return to origin at } j\text{-th instant}]$$

It is to note that $\varphi_1(2m) = 0$ and $\varphi_2(2m-1) = 0$. L is considered to be a multiple of two. Further,

$$\varphi_1(2m+1) = \pi_5(2m+1) - \pi_5(2m-1) ,$$

$$\varphi_2(2m) = \pi_6(2m) - \pi_6(2m-2), \quad \text{for } m=1,2,\dots$$

$\pi_i(k)$ gives the conditional probability. The following transition probabilities are of interest in calculating a limiting state probability P_i . Let h_i denote the first passage probability corresponding to the i -th path in Figure 3.5.

$$h_1 = \text{Prob}[\text{First request for transition } S_L \rightarrow S_{L+1} \text{ without returning to } S_0 (\text{origin}) \text{ during } T_i^0]$$

$$h_2 = \text{Prob}[\text{First request for insertion after the first return to } S_0 \text{ without reaching } S_{L+1} \text{ during } T_i^0]$$

$$h_3 = \text{Prob}[\text{First request for insertion without returning to } S_{L+2} \text{ during } T_i^{L+2}]$$

$$h_4 = \text{Prob}[\text{First request for transition } S_L \rightarrow S_{L+2} \text{ without reaching } S_0 \text{ during } T_i^{L+2}]$$

$$h_5 = \text{Prob}[\text{First request for retrieval without returning to } S_{L+2} \text{ during } T_i^{L+2}]$$

$$h_6 = \text{Prob}[\text{First request for retrieval after the first return to } S_0 \text{ without reaching } S_{L+1} \text{ during } T_i^0]$$

$$h_7 = \text{Prob}[\text{First request for insertion during } T_i^{L+2}] .$$

It can be noticed that

$$\begin{aligned} h_1 &= \sum_{k=0}^{\infty} \varphi_1(2k+1) \cdot \text{Prob}[(2k+1) \text{ or more requests occur} \\ &\quad \text{during } T_i] \\ &= \sum_{k=0}^{\infty} \varphi_1(2k+1) \cdot \left[1 - \sum_{j=0}^{2k} \frac{[(\lambda + \mu)T_i]^j}{j!} e^{-(\lambda + \mu)T_i} \right] \end{aligned}$$

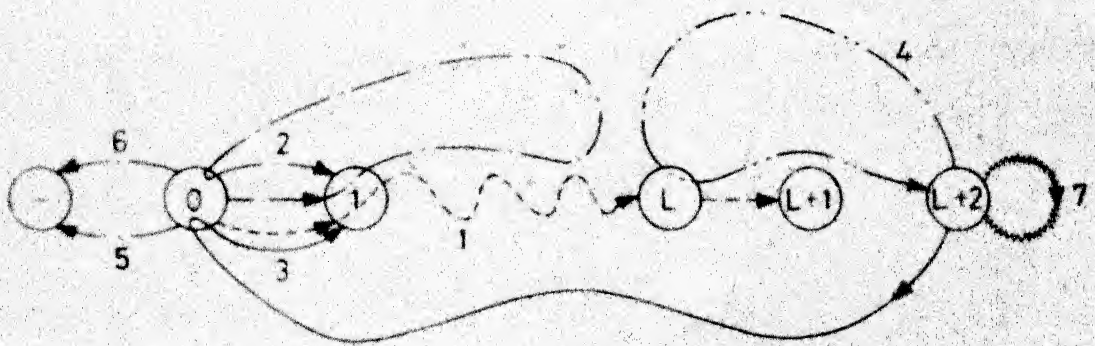


Figure 3.5 TRANSITION PATHS LEADING TO BLOCKING/WAITING STATES

$$\begin{aligned}
 h_2 &= \frac{1}{2} \sum_{k=0}^{\infty} \varphi_2(2k) \cdot \text{Prob}[(2k+1) \text{ or more requests occur} \\
 &\quad \text{during } T_i] \\
 &= \frac{1}{2} \sum_{k=0}^{\infty} \varphi_2(2k) \cdot \left[1 - \sum_{j=0}^{2k} \frac{[(\lambda+\mu)T_i]^j}{j!} e^{-(\lambda+\mu)T_i} \right]
 \end{aligned}$$

$$h_7 = 1/2$$

Further,

$$h_5 = h_3 = h_1,$$

$$h_6 = h_4 = h_2.$$

3.5.3 ESTIMATION OF THE GATING FACTORS

ϵ is considered for the transition $S_0 \rightarrow S_1$ which depends on the probability of reorientation period being over. Reorientation period could be initiated in S_0 or S_{L+2} . We can estimate ϵ as follows

$$\epsilon = (1 - h_2 - h_3 P_{L+2})$$

α is involved in the transition $S_L \rightarrow S_{L+2}$ which depends on the probability of reorientation-period initiated in S_0 or S_{L+2} being over. We can write

$$\alpha = 1 - h_4 P_{L+2} - g_1 P_0$$

where

$$\begin{aligned}
 g_1 &= \text{Prob}[\text{transition } S_L \rightarrow S_{L+1} \text{ occurs}] \\
 &= \pi_5(\infty)
 \end{aligned}$$

For $\beta = 1$,

$$\alpha = 1 - h_4 \frac{\alpha}{1+\alpha} \epsilon P_0 - g_1 P_0$$

$$\alpha^2 + (g_1 + h_4 \epsilon) P_0 \alpha + (g_1 P_0^{-1}) = 0.$$

Solution Procedure:

1. Initial approximation

$$\epsilon = 1 - h_2 ,$$

$$P_0 = 1/(1 + \epsilon L) .$$

- 2.
- $X = \epsilon$

3. Compute
- α
- from

$$\alpha^2 + (g_1 + h_4 \epsilon) P_0 \alpha + (g_1 P_0 - 1) = 0 .$$

- 4.
- $P_{L+2} = \frac{\alpha}{1+\alpha} \epsilon P_0 ,$

$$\epsilon = 1 - h_2 - h_3 P_{L+2} ,$$

$$P_0 = 1/(1 + \epsilon L) .$$

5. If
- $|\epsilon - X| \geq 10^{-6}$
- then goto 2

$$P_n = \epsilon P_0 , \quad \text{for } 1 \leq n \leq L-1$$

$$P_L = \frac{1}{1+\alpha} \epsilon P_0 ,$$

$$P_{L+2} = \alpha P_L .$$

6. Stop.

3.5.4 MEAN ACCESS TIME

Let T_u be the mean access time of an unbuffered stack organization consisting of entirely unidirectional shift-registers.

$$T_u = \frac{1}{2} (N+1) T_s$$

PDF of the time for n -th request is computed from the n -th convolution probability density function of inter-request intervals.

$$H_n(t) = \int_0^t \frac{(2\lambda)(2\lambda y)^{n-1}}{(n-1)!} e^{-2\lambda y} dy, \text{ for } \lambda = \mu$$

Mean Waiting Time for n-th request

$$\bar{W}_n = \int_0^{T_i} (T_i - x) d H_n(x)$$

$$\begin{aligned} \bar{W}_n = T_i & \left[1 - \sum_{j=0}^{n-1} \frac{(2\lambda T_i)^j}{j!} e^{-2\lambda T_i} \right] \\ & - \frac{n}{2\lambda} \left[1 - \sum_{j=0}^n \frac{(2\lambda T_i)^j}{j!} e^{-2\lambda T_i} \right] \end{aligned}$$

Waiting time may arise due to either (a) insertion requests (paths 1, 2, 3, 4, 7) or (b) retrieval requests (paths 5, 6) (see Figure 3.5).

Let x_i be the mean access time in the i-th path

$$x_1 = \sum_{n=L+1}^{\infty} \frac{1}{n} \bar{W}_n \varphi_1(n)$$

$$x_2 = \sum_{m=1}^{\infty} \frac{1}{2(2m+1)} \bar{W}_{2m+1} \varphi_2(2m)$$

$$x_7 = \bar{W}_1.$$

As requests for insertion and retrieval are equiprobable, we can write $x_6 = x_2$. x_1 , x_2 and x_6 are the mean access times as a result of a request waiting during T_i^0 . Similar analysis will yield the mean access times, namely x_3 , x_4 , x_5 and x_7 as a result of a request waiting during T_i^{L+2} . It is to be noted that $x_3 = x_1$, $x_4 = x_2$ and $x_5 = x_3$. Hence,

TABLE 3.1 Performance-improvement (η) of buffered stack memory with variable clock ECM.

$N = 512$

MRT ($\times T_s$)										
L	25	50	75	100	150	200	250	300	350	400
2	69.1	73.9	78.1	81.2	85.4	88.0	89.7	91.0	91.9	92.7
4	80.3	83.4	86.0	88.0	90.7	92.4	93.5	94.3	94.9	95.4
6	85.5	87.8	89.7	91.2	93.2	94.5	95.3	95.9	96.3	96.7
8	88.5	90.3	91.8	93.1	94.7	95.7	96.3	96.8	97.1	97.4
12	91.9	93.1	94.2	95.1	96.3	97.0	97.4	97.8	98.0	98.2
16	93.7	94.7	95.6	96.2	97.1	97.7	98.0	98.3	98.5	98.6
24	95.7	96.4	96.9	97.4	98.0	98.4	98.7	98.8	99.0	99.1

$N = 1024$

2	66.6	69.0	71.5	73.9	78.0	81.2	83.6	85.4	86.8	88.0
4	78.7	80.3	81.9	83.4	86.0	88.0	89.5	90.7	91.6	92.4
6	84.3	85.5	86.7	87.8	89.7	91.2	92.3	93.2	93.9	94.5
8	87.6	88.5	89.4	90.3	91.8	93.1	94.0	94.7	95.2	95.6
12	91.2	91.9	92.5	93.1	94.2	95.1	95.8	96.3	96.7	97.0
16	93.2	93.7	94.2	94.7	95.5	96.2	96.7	97.1	97.4	97.7
24	95.3	95.7	96.0	96.3	96.9	97.4	97.8	98.0	98.2	98.4

$N = 2048$

2	65.3	66.5	67.8	69.0	71.5	73.9	76.1	78.0	79.7	81.2
4	77.9	78.7	79.5	80.3	81.9	83.4	84.8	86.0	87.1	88.0
6	83.7	84.3	84.9	85.5	86.7	87.8	88.8	89.7	90.5	91.2
8	87.1	87.6	88.1	88.5	89.4	90.3	91.1	91.8	92.2	93.1
12	90.8	91.2	91.5	91.9	92.5	93.1	93.7	94.2	94.7	95.1
16	92.9	93.2	93.5	93.7	94.2	94.7	95.1	95.5	95.9	96.2
24	95.1	95.3	95.5	95.7	96.0	96.3	96.7	96.9	97.2	97.4

$$x_5 = x_3 = x_1 ,$$

and $x_6 = x_4 = x_2 .$

We compute the mean access time T_b and the performance-improvement η of the buffered stack as follows:

$$T_b = (x_1 + x_2 + x_6)P_0 + (x_3 + x_4 + x_5)P_{L+2} + \frac{1}{2}x_7 P_{L+2}$$

and

$$\eta = (1 - \frac{T_b}{T_u}) \times 100 \text{ percent} .$$

The analytical values for the performance-improvement are recorded in Table 3.1 for $N = 512, 1024,$ and 2048 .

3.6 VERIFICATION OF THE ABOVE MODEL

A simulation study was carried out in order to verify the analytical results. The program was written in GPSS III and processed on IBM 7044. Table 3.2(a) and (b) compare the analytical and the simulation values for the mean access time, T_b , and the performance-improvement, η , of the buffered stack memories with $N = 512$ and 1024 respectively. The confidence-interval for the simulation results is computed as follows. For a fairly large sample of size S , the variance of the sample mean is estimated as σ^2/S and the distribution of the sample mean can be regarded as normal. Given the sample mean, m , and the sample variance, σ^2 , the 95 percent confidence-interval, c , for the population mean lies between

$m - 1.96 \sigma/\sqrt{S}$ and $m + 1.96 \sigma/\sqrt{S}$. We obtain the mean access time, T_b , and the standard deviation, σ , and compute η and its confidence-interval using the above values. To illustrate with an example, let $T_b = 41.1$, $\sigma = 118.0$, $N = 512$ and $S = 5000$. We can compute performance-improvement, $\eta = (1 - T_b/T_u) \times 100$ percent, that is $\eta = (1 - \frac{41.1 \times 2}{512+1}) \times 100 = 84.0$ percent. The 95 percent confidence-interval for T_b is $(T_b - y)$ and $(T_b + y)$, where $y = 1.96 \times 118.0 / \sqrt{5000}$. Hence η lies between $(\frac{T_u - T_b}{T_u} - \frac{y}{T_u}) \times 100$ and $(\frac{T_u - T_b}{T_u} + \frac{y}{T_u}) \times 100$. We can write $c = y/T_u$ which defines the 95 percent confidence-interval of η . Tables 3.2 and 3.3 record the values for T_b and $(\eta \pm c)$. In the analysis, we assume exponential distribution of mean inter-request times. The analytical results are compared with simulation results in Tables 3.2(a) and (b).

For Erlangian distribution of inter-request times, the simulation study was carried out. Table 3.3 records values for T_b and η for Erlang constant $E = 1, 2$ and 8 , and $N = 512$. It is to be noted that $E = 1$ means exponential distribution and $E = \infty$ a constant distribution.

Figure 3.6 is a plot of performance-improvement versus buffer-length for $N = 512$ and $MRT = 100T_s$, $200T_s$ and $400T_s$. Solid lines correspond to the simulation results, whereas dashed lines to the analytical results. Figure 3.7 is a plot of η versus MRT for $L = 4, 8$ and 12 .

TABLE 3.2(a) Buffered stack with variable clock ECM.
Comparison of analytical and simulation
results. $N = 512$, $E = 1$.

SIMULATION						ANALYTICAL					
MRT (x T_s)	L	T_b	$\eta \pm c$	T_b	η	MRT (x T_s)	L	T_b	$\eta \pm c$	T_b	η
25	2	122.5	52.4 \pm 2.2	79.2	69.1	50	2	100.5	61.0 \pm 1.9	66.8	73.9
	4	68.5	73.5 \pm 1.7	50.3	80.3		4	56.6	78.2 \pm 1.5	42.4	83.4
	8	35.5	86.4 \pm 1.3	29.3	88.5		8	28.4	89.1 \pm 1.1	24.8	90.3
	12	27.5	89.5 \pm 1.2	20.8	91.9		12	28.5	91.1 \pm 0.9	17.5	93.1
	16	19.8	92.6 \pm 1.0	16.1	93.7		16	15.0	94.1 \pm 0.9	13.6	94.7
75	2	84.9	67.7 \pm 1.7	56.2	78.1	100	2	73.8	71.5 \pm 1.6	48.0	81.2
	4	48.4	81.3 \pm 1.4	35.8	86.0		4	41.1	84.0 \pm 1.3	30.6	88.0
	8	22.3	91.5 \pm 1.0	20.9	91.8		8	20.3	92.2 \pm 1.0	17.8	93.1
	12	17.0	93.8 \pm 0.9	14.7	94.2		12	14.1	94.6 \pm 0.8	12.5	95.1
	16	13.3	95.0 \pm 0.8	11.4	95.5		16	11.7	95.8 \pm 0.8	9.7	96.2
150	2	50.7	80.2 \pm 1.4	37.3	85.4	200	2	38.2	85.2 \pm 1.1	30.8	88.0
	4	31.0	87.9 \pm 1.1	23.7	90.7		4	22.1	91.5 \pm 0.9	19.5	92.4
	8	17.0	93.8 \pm 0.8	13.6	94.7		8	13.8	94.6 \pm 0.7	11.1	95.7
	12	11.7	95.8 \pm 0.7	9.5	96.3		12	9.1	96.5 \pm 0.6	7.8	97.0
	16	10.0	96.1 \pm 0.6	7.3	97.1		16	5.9	97.8 \pm 0.5	6.0	97.7
300	2	29.4	88.7 \pm 1.0	23.1	91.0	400	2	20.2	92.1 \pm 0.9	18.7	92.7
	4	17.5	93.2 \pm 0.8	14.5	94.3		4	13.2	95.0 \pm 0.7	11.7	95.4
	8	9.5	96.5 \pm 0.6	8.2	96.8		8	6.9	97.4 \pm 0.6	6.6	97.4
	12	6.0	97.7 \pm 0.5	5.7	97.8		12	3.0	98.9 \pm 0.3	4.6	98.2
	16	5.0	98.0 \pm 0.4	4.4	98.3		16	2.9	98.9 \pm 0.3	3.5	98.6

MRT = Mean inter-request time, L = Length of buffer (fast stack),
 T_b = Mean access time, η = Performance-improvement,
 c = Confidence-interval.

TABLE 3.2(b) Buffered stack with variable clock ECM.
Comparison of analytical and simulation
results. $N = 1024$, $E = 1$.

SIMULATION						ANALYTICAL						SIMULATION						ANALYTICAL					
MRT ($\times T_s$)	L	T_b	$\eta \pm c$	T_b	η	MRT ($\times T_s$)	L	T_b	$\eta \pm c$	T_b	η	MRT ($\times T_s$)	L	T_b	$\eta \pm c$	T_b	η	MRT ($\times T_s$)	L	T_b	$\eta \pm c$	T_b	η
25	2	270.0	47.3 \pm 2.2	171.2	66.6	50	2	245.0	52.4 \pm 2.2	158.5	69.0	75	2	212.8	58.6 \pm 2.0	145.9	71.5	100	2	195.6	62.0 \pm 1.9	133.7	73.9
	4	163.7	68.2 \pm 1.9	109.0	78.7		4	138.3	73.1 \pm 1.7	100.7	80.3		4	128.2	75.0 \pm 1.6	92.6	81.9		4	108.7	79.0 \pm 1.5	85.9	83.4
	8	79.0	86.4 \pm 1.3	63.7	87.6		8	70.9	86.4 \pm 1.4	58.7	88.5		8	72.2	85.8 \pm 1.3	54.0	89.4		8	58.4	88.7 \pm 1.2	49.6	90.3
	12	60.8	88.3 \pm 1.3	45.0	91.2		12	51.4	90.1 \pm 1.1	41.6	91.9		12	60.9	88.3 \pm 1.2	38.3	92.5		12	47.0	91.1 \pm 1.0	35.1	93.1
	16	48.8	90.3 \pm 1.1	34.9	93.2		16	40.4	92.2 \pm 1.0	32.2	93.7		16	32.5	93.8 \pm 0.9	29.6	94.2		16	25.3	95.2 \pm 0.8	27.1	94.7
150	2	165.2	67.8 \pm 1.7	112.5	78.0	200	2	149.7	70.7 \pm 1.5	96.2	81.2	300	2	97.6	80.8 \pm 1.3	74.8	85.4	400	2	76.4	85.2 \pm 1.2	61.7	88.0
	4	98.2	80.9 \pm 1.5	71.7	86.0		4	78.0	85.0 \pm 1.3	61.4	88.0		4	57.9	88.9 \pm 1.1	47.5	90.7		4	92.6	89.9 \pm 1.0	39.0	92.4
	8	45.0	91.7 \pm 1.0	41.8	91.8		8	40.7	92.2 \pm 0.9	35.6	93.0		8	34.0	93.8 \pm 0.9	27.3	94.7		8	28.2	94.6 \pm 0.8	22.3	95.7
	12	34.0	93.8 \pm 0.9	29.5	94.2		12	29.8	94.4 \pm 0.8	25.1	95.1		12	27.0	94.8 \pm 0.7	19.1	96.3		12	18.2	96.5 \pm 0.7	15.6	97.0
	16	26.4	95.0 \pm 0.8	22.8	95.5		16	23.4	95.8 \pm 0.8	19.3	96.2		16	20.2	96.1 \pm 0.6	14.7	97.1		16	15.6	97.1 \pm 0.6	11.9	97.7

MRT = Mean inter-request time, L = Length of buffer (fast stack),
 T_b = Mean access time, η = Performance improvement,
 c = Confidence interval.

TABLE 3.3 Effect of variation in Erlang constant (E) on the performance of a buffered stack with variable clock BCM. N = 512.

MRT (x T _s)	L	E = 1		E = 2		E = 8	
		T _b	$\eta \pm c$	T _b	$\eta \pm c$	T _b	$\eta \pm c$
100	2	73.8	71.5 \pm 1.6	72.2	71.9 \pm 1.5	70.0	72.6 \pm 1.2
	4	41.1	84.0 \pm 1.3	37.6	85.6 \pm 1.2	36.0	86.0 \pm 1.1
	8	20.0	92.2 \pm 1.0	19.6	92.3 \pm 1.0	19.0	92.6 \pm 0.8
	12	14.1	94.6 \pm 0.8	13.2	95.0 \pm 0.7	12.6	95.1 \pm 0.5
200	2	38.2	85.2 \pm 1.1	36.8	85.6 \pm 1.1	34.8	86.8 \pm 0.9
	4	22.1	91.5 \pm 0.9	18.5	92.8 \pm 0.8	18.1	93.0 \pm 0.6
	8	13.8	94.6 \pm 0.7	13.5	94.7 \pm 0.7	13.0	94.9 \pm 0.5
	12	9.1	96.5 \pm 0.6	8.7	96.9 \pm 0.6	8.0	96.9 \pm 0.4
300	2	29.4	88.7 \pm 1.0	21.6	91.8 \pm 0.8	16.4	93.8 \pm 0.6
	4	17.5	93.2 \pm 0.8	12.6	95.4 \pm 0.7	10.6	95.9 \pm 0.5
	8	9.5	96.5 \pm 0.6	8.4	96.9 \pm 0.5	6.6	97.7 \pm 0.3
	12	6.0	97.7 \pm 0.5	6.0	97.7 \pm 0.4	3.7	98.9 \pm 0.3
400	2	20.2	92.1 \pm 0.9	14.3	94.4 \pm 0.7	12.4	95.4 \pm 0.6
	4	13.2	95.0 \pm 0.7	11.1	95.7 \pm 0.6	8.5	96.9 \pm 0.5
	8	6.9	97.4 \pm 0.6	4.6	98.2 \pm 0.4	3.1	98.9 \pm 0.3
	12	3.0	98.9 \pm 0.3	2.8	99.0 \pm 0.3	2.4	99.0 \pm 0.2

MRT = Mean inter-request time, L = Length of buffer
(fast stack)

T_b = Mean access time, η = Performance improvement

c = Confidence interval.

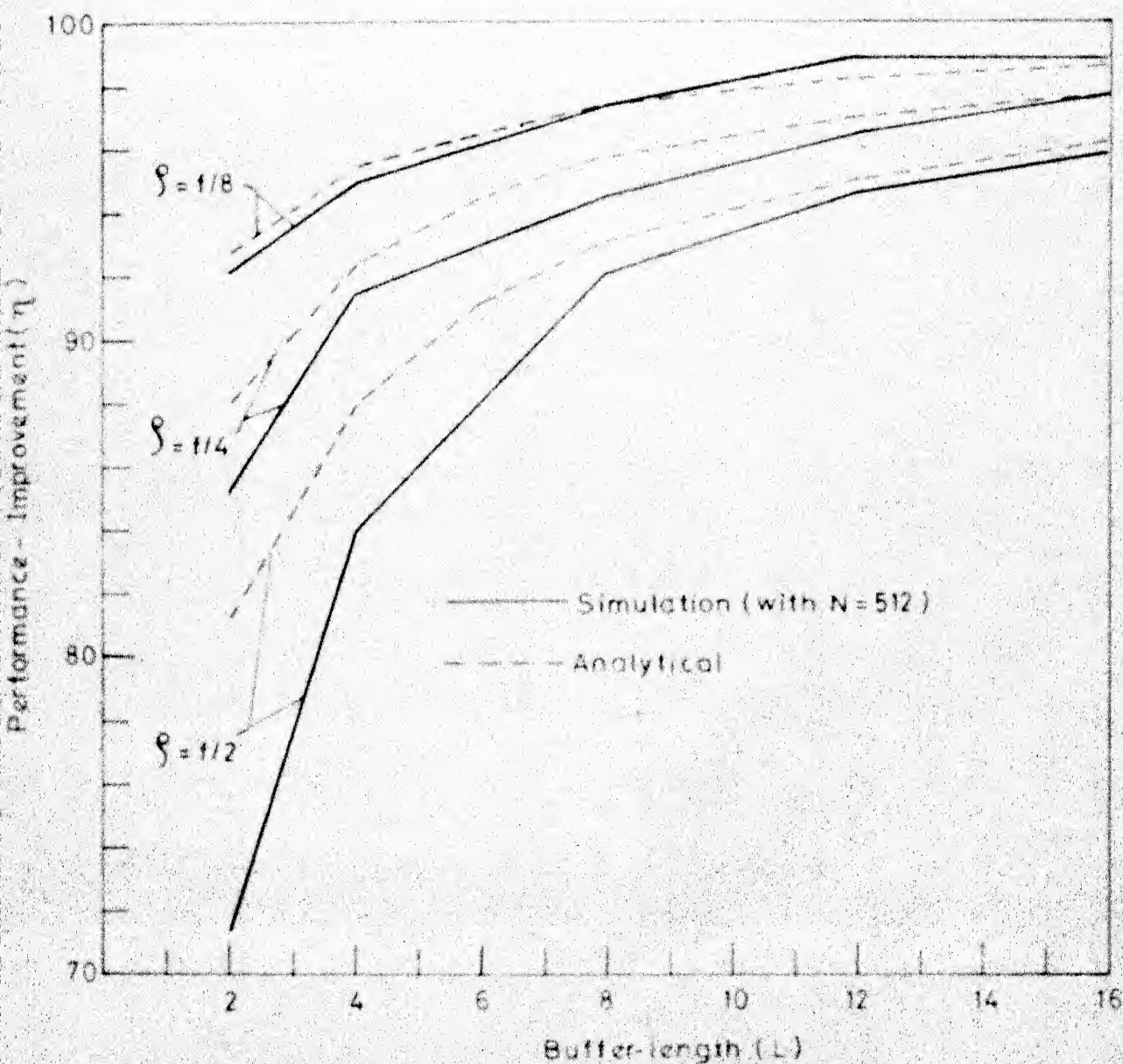


Figure 3.6 BUFFERED STACK MEMORY WITH VARIABLE CLOCK ECM.
- η VERSUS L .

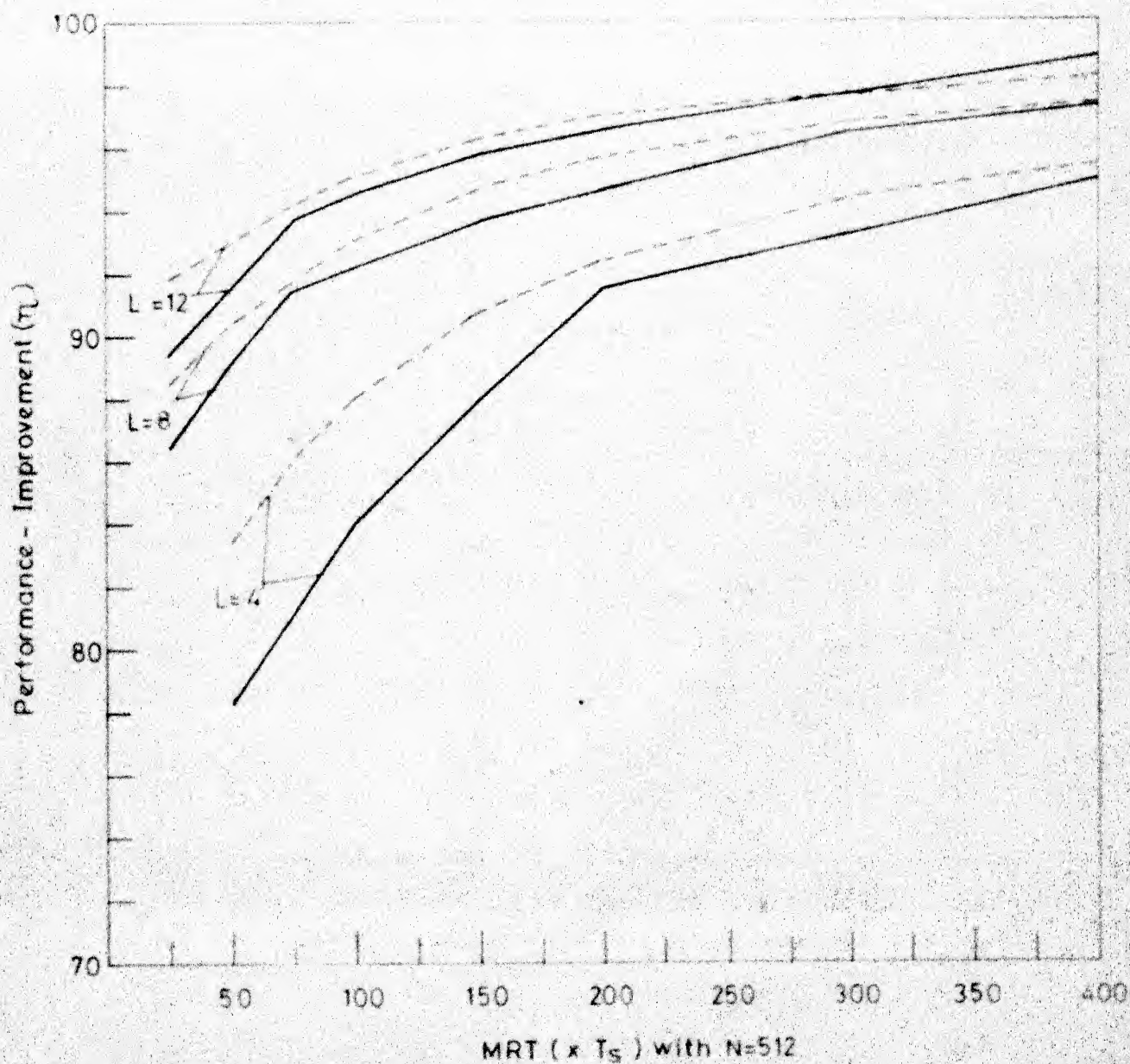


Figure 3.7 BUFFERED STACK MEMORY WITH VARIABLE CLOCK ECM.
- η VERSUS MRT.

3.7 DISCUSSION OF RESULTS

Table 3.1 serves as a design guide table. To illustrate with an example, a buffer of length 8 with a shift-register stack of size 1024 exhibits 90.3 percent improvement in performance over an unbuffered SR-stack of the same size for the mean inter-request time of $100T_s$.

We observe in Table 3.1 that η depends on the product (ρ) of mean arrival rate, γ , and the reorientation time, NT_s , that is $\rho = \gamma NT_s$. We define a constant f equal to 10.24. ρ can be expressed in terms of f . ρ in terms of f is recorded in Table 3.4 for a few pairs of N and MRT. For a given L , η is identical for ($N = 512$ and $MRT = 100T_s$), and ($N = 1024$ and $MRT = 200T_s$) as $\rho = \frac{1}{2}f$ for both of these pairs.

TABLE 3.4
 ρ in terms of f

$N \searrow \xrightarrow{MRT(xT_s)}$	25	50	75	100	150	200	300	400
512	$2f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$	$\frac{1}{3}f$	$\frac{1}{4}f$	$\frac{1}{6}f$	$\frac{1}{8}f$
1024	$4f$	$2f$	$\frac{4}{3}f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$	$\frac{1}{3}f$	$\frac{1}{4}f$
2048	$8f$	$4f$	$\frac{8}{3}f$	$2f$	$\frac{4}{3}f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$

Analytical results agree well with those obtained from the simulation. The access time decreases exponentially with an increase in the size of buffer. The lower the value of ρ the higher is the value of η (refer to Figure 3.6).

Consideration of Erlangian distribution of inter-request times shows that an increase in the value of Erlang constant, E , results in better performance. However, the increase in η is not appreciable. The confidence-interval shrinks with an increase in Erlang constant.

3.8 APPLICATIONS

Typical application areas of buffered stack include language translation; subroutine operations; recursive operations; dynamic programming; and processing of object programs written in a structured language, like ALGOL.

3.9 CONCLUSION

A scheme for a buffered stack memory with variable clock ECM (e.g., static shift-registers) is proposed. An addition of a small size buffer (fast stack) to a large size ECM improves its performance drastically. In order to quantify the improvement, we define the performance-improvement (η) of a buffered stack over an unbuffered stack. A fast stack of size 12 yields the performance-improvement of more than 95 percent with an ECM of size 512 for $MRT \geq 100T_s$.

CHAPTER 4

A NOVEL TECHNIQUE FOR MULTIPLE INSERTIONS AND DELETIONS

A scheme is proposed in this chapter to facilitate multiple insertions and deletions in an ECM. The property of information-mobility is exploited to perform these operations.

4.1 INTRODUCTION

Information moves in an Electronic Cyclic Memory (ECM). An information-channel within the device remains fixed in CCDs and MOS shift-registers, whereas it can be varied in magnetic bubbles. This additional flexibility facilitates operations, like global shift, detached shift, exchange and delta-exchange in a Magnetic Bubble Memory. These are the basic operations required to perform dynamic data rearrangement [C. Tung et.al., 1975; T.C.Chen and C. Tung, 1976].

Mobile information bits can be routed through an external buffer whose length can be varied by means of a combinatorial circuit. Thus it is possible to expand or contract a data path. An insertion can be performed by expanding the data path. Similarly, a deletion can be performed by contracting the data path. Garbage collection [P.W.Abraham, 1968] is a special case of the deletion operation and can be performed during each cycle of rotation.

Hoff and Mazor [1971] propose an elementary scheme for adding and deleting data in a shift-register memory. A single bit can be inserted within a record during a cycle of rotation. The deletion operation results in packets of vacant positions. We propose a scheme which enables multiple insertions to be performed during a cycle. Multiple deletions can be performed without creating packets of vacant positions.

Such a scheme will be widely applicable in non-numeric processing.

4.2 DESCRIPTION OF THE SCHEME

A schematic diagram of an ECM-Organization for multiple insertions and deletions is shown in Figure 4.1. We use the terms ECM and SR interchangeably.

($m+1$) N -bit shift-registers (SRs) in parallel provide m -bit word together with one activity bit which is set to 1 when meaningful information is stored in the corresponding location, else it is set to 0 to indicate a vacant position. N may range from 512 to 2048. A small size buffer is associated with each SR. Such a buffer consists of a Parallel-In-Serial-Out (PISO) shift-register, typically 8-32 bits long.

There is a control unit or an inexpensive micro-processor [E.A. Torrero, 1975] which compares m -bit word from SRs with the content of the comparand register (CR)

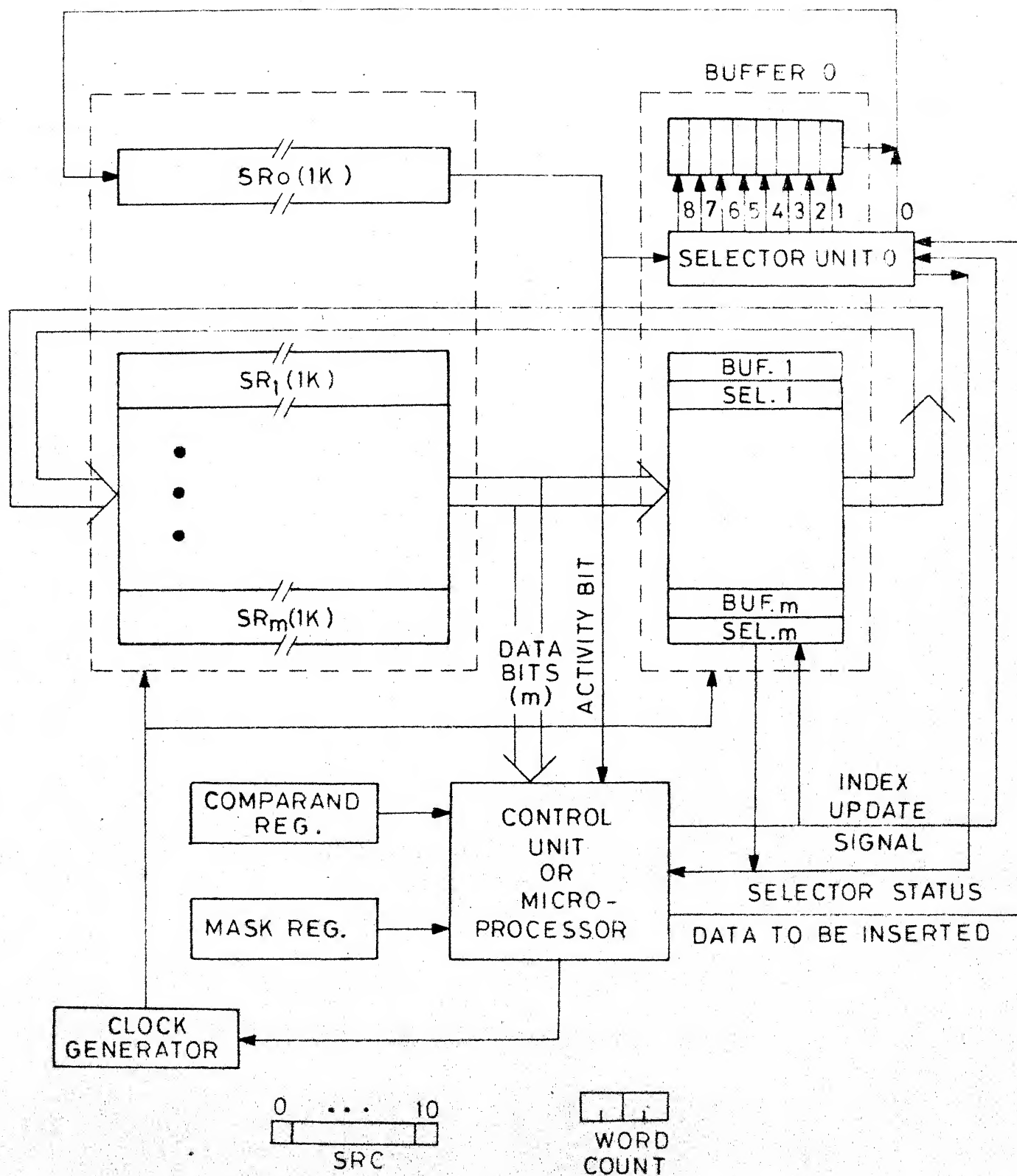


FIG.4.1 ECM ORGANIZATION FOR NON-NUMERIC PROCESSING.

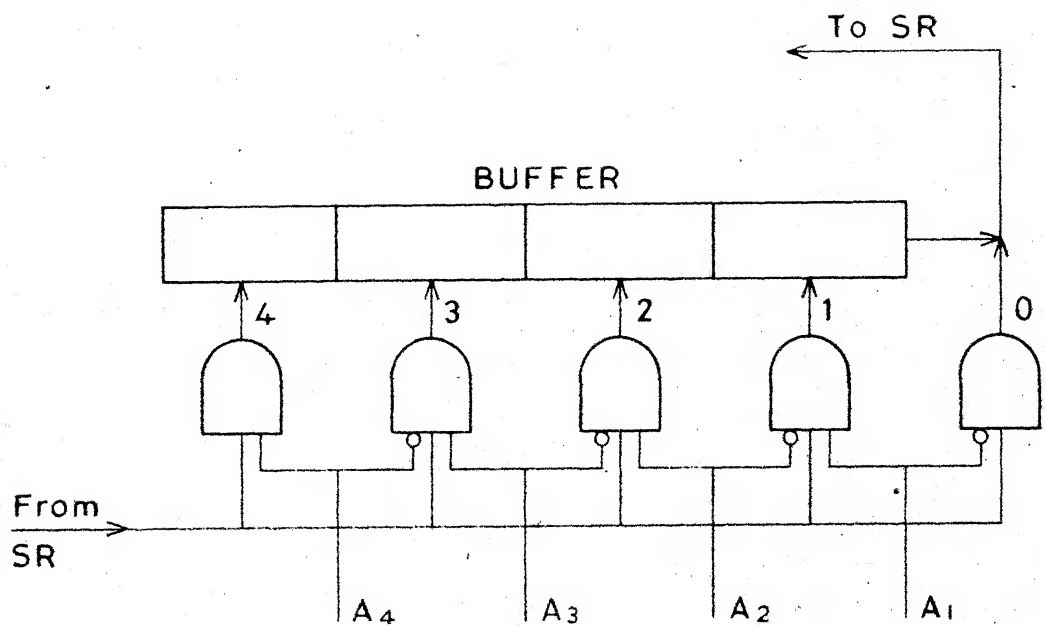


Figure 4.2 SELECTOR UNIT

sets highest index pointing
to non-vacant position.

and issues a command for insertion or deletion of a string. A mask register can be used to mask certain bits while performing comparison or any other logical operation.

Bits from SR enter the buffer through a selector unit (Figure 4.2). A selector unit is a combinatorial circuit to connect the SR-output to the buffer by selecting the highest index pointing to a non-vacant position. It can increment or decrement the switch index by suitably modifying the activity bits so as to expand or contract the data path via buffer. It can selectively store information bits into the available positions to the left of the currently activated switch. All the selector units are synchronized and perform an operation simultaneously. 'Increment (decrement) a switch index', and 'Insert a string of information bits' are typical control commands which reach a selector unit.

4.3 OPERATION

Refer to Figure 4.3. Switch index is initialized to $L/2$, where L is the length of the buffer. Data path in the buffer is $L/2, \frac{L}{2} - 1, \dots, 1$. Locations $(\frac{L}{2} + 1), \dots, L$ are unused and store all zeros. Switch index is decremented by one for a deletion operation. Suppose k is the index of the currently activated switch, $(L-k)$ unused left positions are available for consecutive insertions and k positions are occupied by information bits. In order to insert b information bits after a specified location, $\min(b, (L-k))$

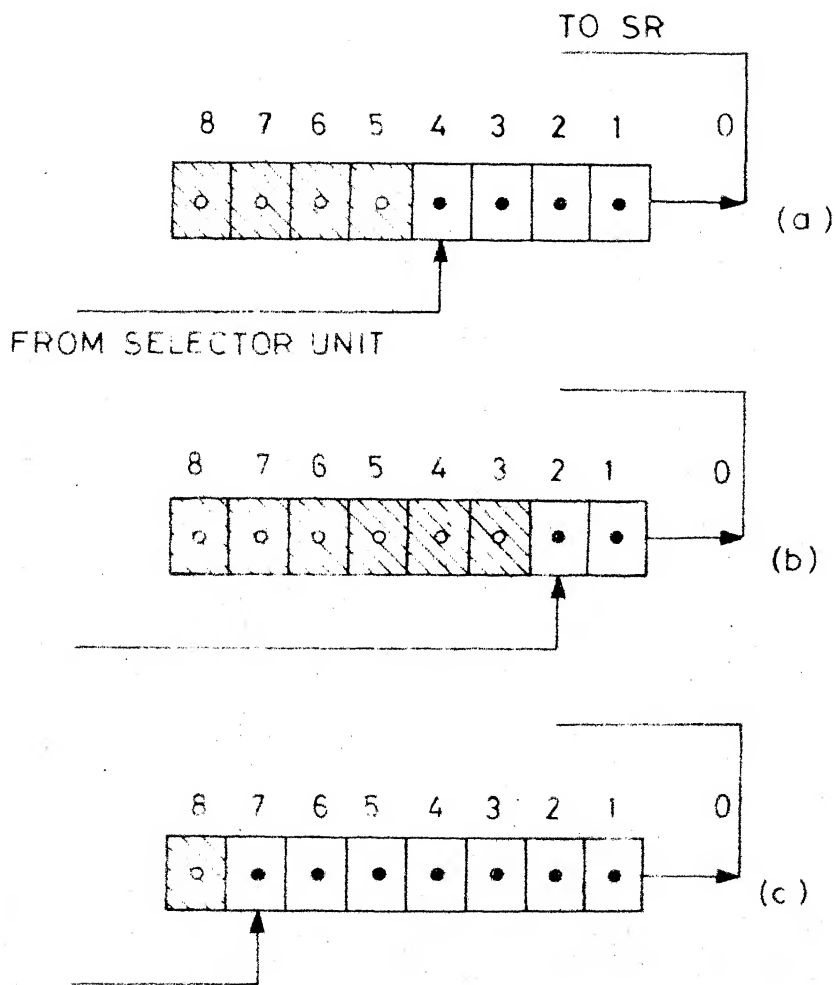


FIG. 4.3 STATES OF BUFFER AFTER (a) INITIALIZATION, (b) DELETION OF A STRING, (c) INSERTION OF A STRING.

are stored in the buffer and the switch index is set to the leftmost information bit in the buffer. Remaining information bits to be inserted (if any) wait for the next cycle. Similarly $\min(b, k)$ can be deleted during the consecutive shift periods and the remaining ones wait for the next cycle. b is the length of information to be deleted.

At the end of the file the switch index is initialized to $L/2$. If the index is greater than $L/2$, it is decremented at every occurrence of a shift clock until $L/2$ and a zero bit is stored in the unused vacant position of the buffer. If the index is less than $L/2$, the switch is set to $L/2$ and vacant positions are added to the end of the file. This is a 'garbage collection'.

Sequence of operations, namely insertion and deletion of a string to be performed on a file can be categorized as:

S_a : a sequence of operations in which an operation does not depend on previous ones.

S_b : a sequence of operations in which an operation is affected by the execution of previous operations.

In the case of S_a , incomplete operations are stored to be scanned during the next cycle. Operations to be performed on a file may be stored in a lookaside memory.

In the case of S_b , an operation cannot be performed if the previous operation is yet to be completed.

To modify a location, desired data is written into that location by successively storing the desired information bits.

ECM, indeed, forms a bit-parallel and word-serial associative memory with a data transfer rate higher than what could be obtained from a RAM of equivalent price. Further, the ability of an ECM to operate over a wide range of transfer rates imparts a great deal of flexibility to perform string manipulative operation.

4.4 DESIGN METHODOLOGY

Let λ and μ be the mean arrival rates of requests for insertion and deletion respectively. There may be a request for multiple insertions or deletions. In order to determine the length of buffer associated with an ECM, we consider the uniform distribution of length of insertions and deletions with ℓ_i and ℓ_d as their respective mean burst-lengths. Further, we assume that the location at which an insertion or a deletion operation is to be performed is uniformly distributed over the entire cycle. Hence the mean access time is equal to $\frac{1}{2}NT_s$ for an operation is performed to completion within the same cycle. With a finite length of buffer, an operation may require a few more cycles.

A queue will build up if the mean service time is greater than the mean inter-arrival time. Thus the mean service time becomes the design criterion. It is desirable that a request should be serviced within at most two cycles. This poses a constraint,

$$1 + 2(\bar{b} - 1) \leq L/2,$$

where $\bar{b} = \max(\ell_i, \ell_d)$. Let $u = 1 + 2(\bar{b} - 1)$. The length of insertions and deletions is considered to be uniformly distributed over an interval $(1, u)$. In order to determine the mean service time, we consider the requests for insertions. Similar analysis holds for requests for deletions. Let P_k be the probability of k information bits being in the buffer. We obtain the mean service time,

$$\bar{x} = \frac{1}{2} NT_s + \sum_{\ell=1}^u \sum_{k=L-\ell+1}^L P_k Z(\ell, k) \frac{1}{u} NT_s$$

where, $Z(\ell, k)$ is the smallest integer greater than or equal to $(\frac{\ell - (L-k)}{L/2})$. It is to be noted that

$$Z(\ell, k) = 1 \quad \text{for } u \leq L/2.$$

A simulation study of the proposed scheme was carried out with our original assumptions that the location and the length of insertions and deletions follow uniform distribution. The ratio of the mean service time to the cyclic time, \bar{x}/NT_s is tabulated in Table 4.1 for $L = 2, 4, 8$ and 16 ; and $\bar{b} = 1, 1.5, 2.0, 2.5, 3.0$ and 4.0 . The ratio \bar{x}/NT_s is found to be same for $N = 512, 1024$, and so on.

TABLE 4.1 Mean service time in a buffered ECM.
Comparison of analytical results with
simulation results.

\bar{b}	Burst Interval	L	SIMULA- TION \bar{x}/NT_s	ANALY- TICAL \bar{x}/NT_s	\bar{b}	Burst Inter- val	L	SIMULA- TION \bar{x}/NT_s	ANALY- TICAL \bar{x}/NT_s
1	1	2	0.84	0.83	1.5	(1-2)	2	1.23	1.17
		4	0.63	0.70			4	0.78	0.80
		8	0.55	0.61			8	0.61	0.67
		16	0.52	0.56			16	0.55	0.59
2.0	(1-3)	2	1.65	1.61	2.5	(1-4)	2	2.10	2.08
		4	0.96	0.97			4	1.17	1.15
		8	0.67	0.72			8	0.75	0.78
		16	0.57	0.62			16	0.59	0.65
3.0	(1-5)	2	2.56	2.57	4.0	(1-7)	2	3.57	3.55
		4	1.38	1.38			4	1.83	1.84
		8	0.84	0.86			8	1.03	1.04
		16	0.63	0.68			16	0.70	0.735

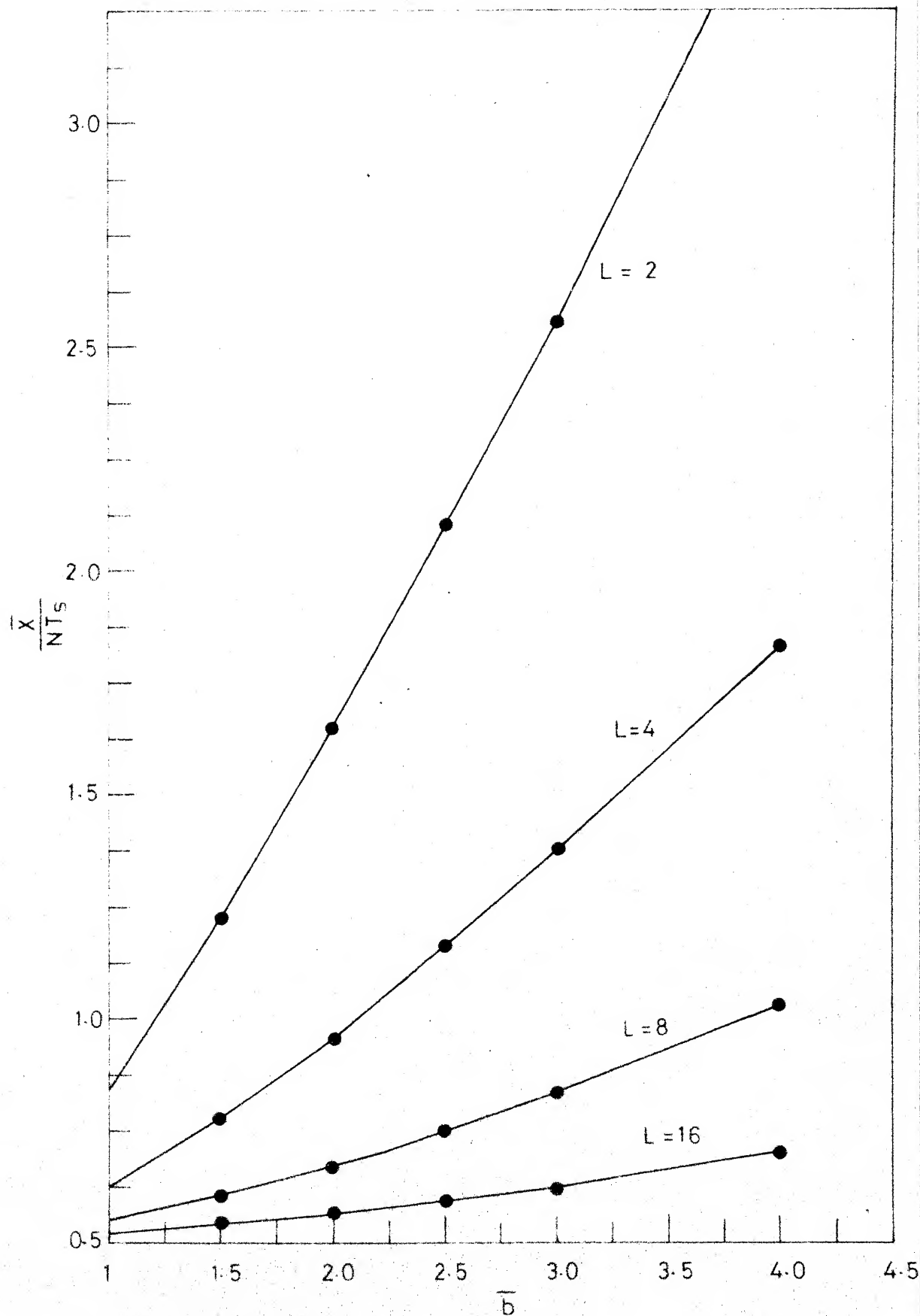


Figure 4.4 \bar{x}/NT_s VERSUS MEAN BURST-LENGTH (\bar{b}).

Figure 4.4 shows the plots of T_b/NT_s versus \bar{b} for $L = 2, 4, 8$ and 16 .

Computation of \bar{x} can be simplified with the assumption of equal state-probabilities. We obtain

$$\bar{x} = \frac{1}{2} NT_s + \sum_{\ell=1}^u \sum_{k=L-\ell+1}^L \frac{Z(\ell, k)}{L+1} \frac{1}{u} NT_s$$

To illustrate with an example, we consider $L = 4$ and $\bar{b} = 1.5$. Hence $u = 1+2(1.5-1) = 2$ and

$$\begin{aligned} \bar{x} &= \frac{1}{2} NT_s + \sum_{\ell=1}^2 \sum_{k=4-\ell+1}^4 \frac{Z(\ell, k)}{5} \frac{1}{2} NT_s \\ &= \frac{1}{2} NT_s + \frac{1}{10}(1 + (1+1)) \\ &= 0.80 NT_s \end{aligned}$$

The corresponding simulation value of \bar{x} is $0.78 NT_s$. The estimated value of $\frac{\bar{x}}{NT_s}$ are compared with the corresponding simulation values in Table 4.1. They agree well.

4.5 APPLICATIONS

Some applications that might benefit from the ability of the proposed scheme to perform string-manipulative operations include interactive terminals, information storage and retrieval, symbol manipulation and evaluation of recursive functions.

Higher-level instructions for string-manipulation are most desirable. These might include, TMI (Test with Mask and Insert), TMD (Test with Mask and Delete) instructions and other instructions for easy testing and modification of fields within data words in the buffer.

4.6 CONCLUSION

A buffered ECM organization has been proposed to perform multiple insertions and deletions. An analytic method is suggested to determine the buffer-length for a given mean burst-length. It is concluded that a buffer of size 16 yields the mean service time of less than 0.75 cycle for mean burst length ≤ 4.0 .

CHAPTER 5

BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM

In this chapter a new scheme is proposed for a buffered stack memory organization. The earlier scheme discussed in Chapter 3 consists of ECMs with variable clock rate, whereas in this scheme the clock rate of ECMs is considered to be constant. We refer to a dynamic shift-register as a constant clock ECM. Addition of a few registers on top of shift-register stack (SS), dramatically, improves the access time. Buffer consisting of these registers is periodically initialized to half of its capacity so as to facilitate fast insertions and retrievals during the recirculation period of SS.

5.1 INTRODUCTION

A stack is a memory with Last-In-First Out (LIFO) access discipline. Use of a stack facilitates recursion, enables fast interrupt handling, increases speed of computation by virtue of code compression which is inherited from its implicit addressing [C.B. Carlon, 1963; E.I. Organick, 1973].

In Chapter 3, we discussed a buffered stack memory organization with variable clock ECMs. In this chapter, we propose a new scheme for a buffered stack memory consisting of a small size fast stack (FS) and large

size ECMs with constant clock rate, e.g., dynamic shift-registers. We use the terms ECM and SR interchangeably.

G. Philokyprou and A. Zacharakopoulos [1968] give a design of a stacking register. It is called a bubble-up register as a data word propagates through the register. In our scheme, the design of a fast stack employs the technique of multiple insertions and deletions in an ECM as given in Chapter 4. With the use of selector units, the content of the fast stack is initialized periodically (loosely speaking) to half of its capacity.

5.2 DESCRIPTION OF THE SCHEME

A buffered stack memory consists of m N -bit SRs (ECMs) in parallel and a fast stack of size L . Typical values for N may be 512-2048 and for L 4-32, m corresponds to the size of a word. The clock rate to ECM is considered to be constant. Figure 5.1 depicts a single bit configuration of such a buffered stack memory with constant clock ECMs. The fast stack (FS) consists of high speed registers. We refer to the upper half of FS as B_1 and the lower half as B_2 . Associated in this scheme is a Parallel-In-Serial-Out (PISO) shift register, B_0 of a size equal to that of B_2 . SELECTOR-0 is the selector unit associated with B_0 . This selects and dynamically sets the highest switch index pointing to a nonvacant position in B_0 . If B_0 is empty its switch index-0 is set which connects it to the input of SR. Output of SR is an input to SELECTOR-1

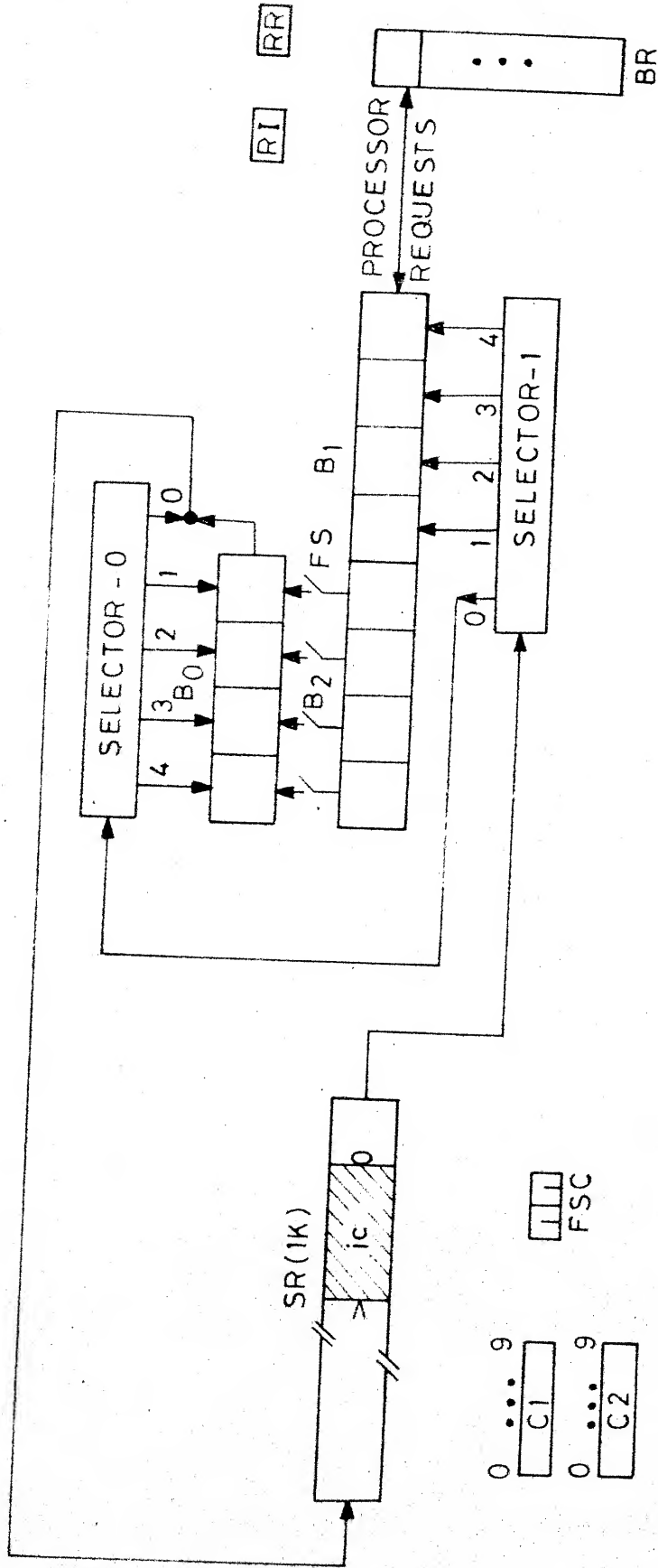
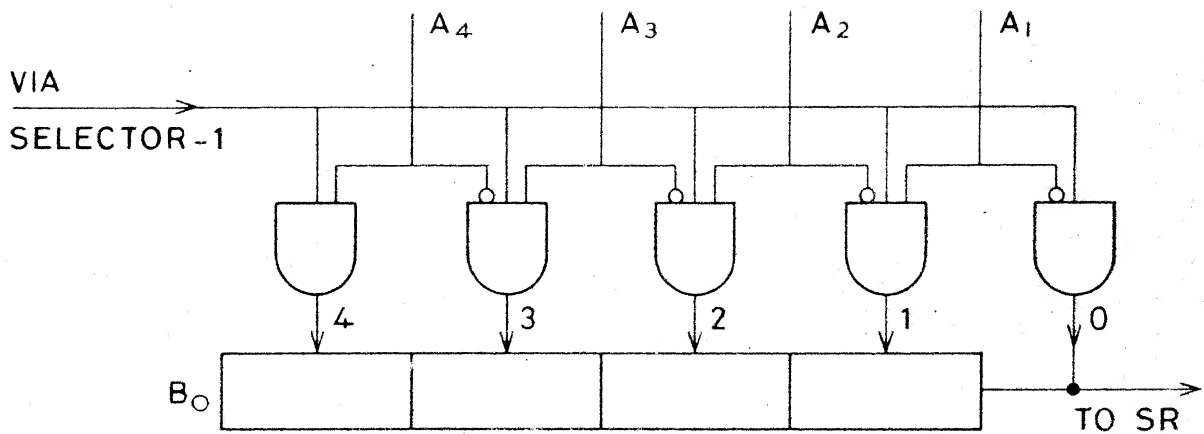
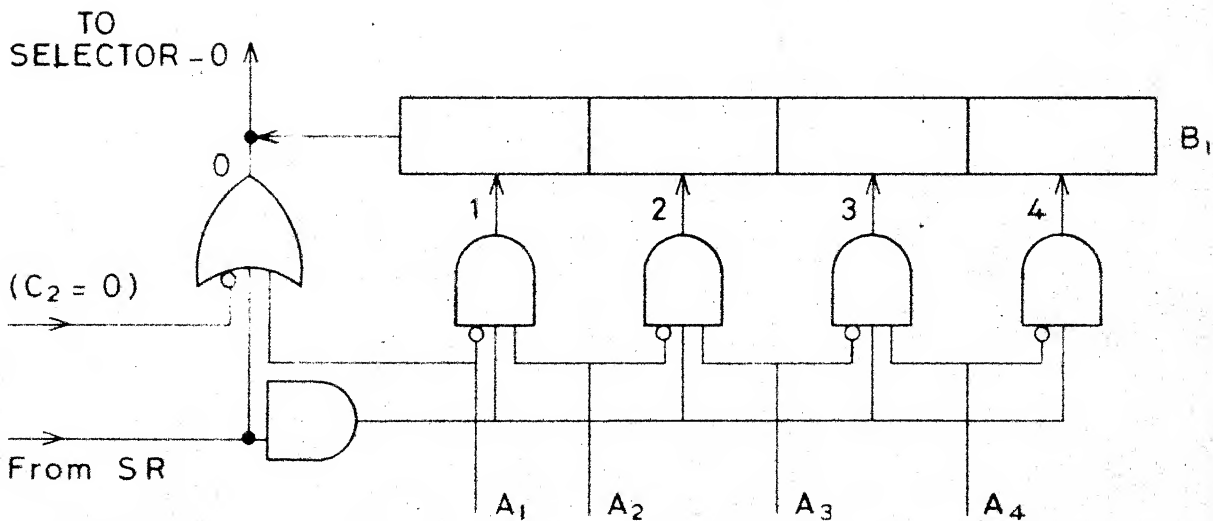


FIG. 5.1 BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM



(a) SELECTOR-0

sets highest index pointing
to non-vacant position.



(b) SELECTOR-1

sets highest index pointing
to vacant position

Figure 5.2 SELECTOR UNITS.

associated with B_1 . This selector unit dynamically sets the highest switch index pointing to a vacant position in B_1 . If B_1 is full, its switch index-0 is set which connects it to the input of SELECTOR-0. Counter FSC indicates the current content of FS. Counter C_1 records distance of the tail of ic where C_2 records distance of the head of ic from the output port of SR. The schematic diagrams of the selector units are shown in Figure 5.2.

5.3 OPERATION

Information stored in SR recirculates via SELECTOR-1 and SELECTOR-0. Buffer is initialized to half of its capacity periodically (loosely speaking). As the head of the information chunk (ic) reaches the output port of SR, SELECTOR-1 selectively transfers the top of the ic through the highest switch index to the vacant position in B_1 . Transfer takes place during the consecutive shift periods until B_1 is full. Then information flows via SELECTOR-0 back to the input port of SR. If B_2 is non-empty, the content of B_2 is transferred in parallel into B_0 and B_2 is left empty and becomes available for possible insertions during the recirculation period. Depending on the occupancy of B_0 SELECTOR-0 sets the highest switch index pointing to a nonvacant position in B_0 :

A request for insertion waits if FS is found full. Similarly a request for retrieval waits if FS is found empty.

5.4 MATHEMATICAL MODEL

In order to obtain the probability of buffer-underflow and overflow, a Markov process containing two absorbing states is developed. This is illustrated by a specific example. Refer to Figure 5.3. Let $L = 4$, E_{-1} and E_5 are the absorbing states. E_{-1} corresponds to the state when a request for Retrieval occurred after the fast stack (FS) became empty during the recirculation period. E_5 corresponds to the state when a request for insertion occurred after FS became full during the recirculation period. Buffer is initialized to E_2 .

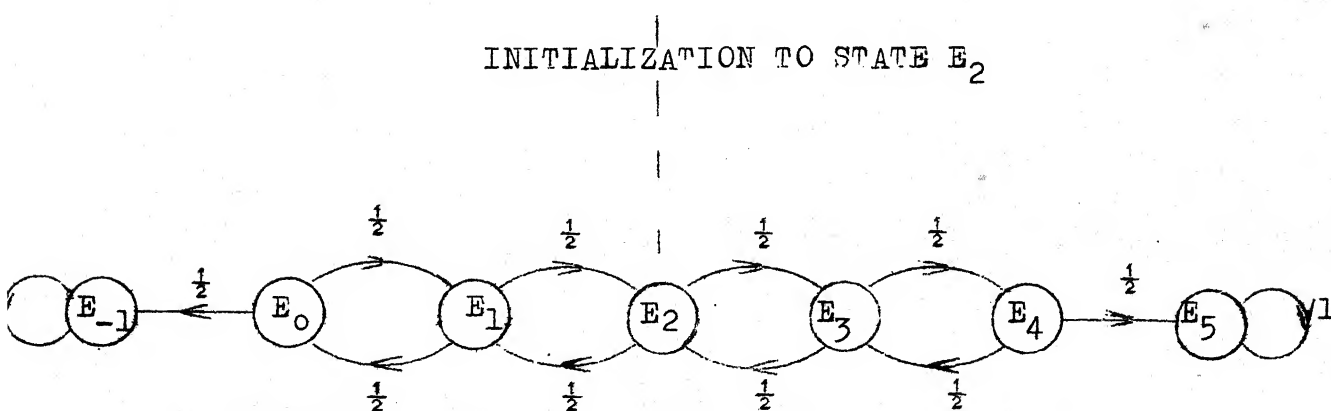


Figure 5.3 STATE-TRANSITION DIAGRAM

Transition probability matrix M can be written as shown below:

$$M = \begin{array}{c|cccccc|c} & E_{-1} & E_0 & E_1 & E_2 & E_3 & E_4 & E_5 & \\ \hline & 1 & 0 & 0 & 0 & 0 & 0 & 0 & E_{-1} \\ & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & E_0 \\ & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & E_1 \\ & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & E_2 \\ & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & E_3 \\ & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & E_4 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 1 & E_5 \end{array}$$

Let $\pi(n)$ be the probability vector at the n-th instant. Following recurrence relation can be used to calculate the probability vectors.

$$\pi(n+1) = \pi(n).M$$

At the instant of initialization, the probability vector is

$$\pi(1) = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0)$$

Subsequent probability vectors are,

$$\pi(2) = (0 \quad 0 \quad 1/2 \quad 0 \quad 1/2 \quad 0 \quad 0)$$

$$\pi(3) = (0 \quad 1/4 \quad 0 \quad 1/2 \quad 0 \quad 1/4 \quad 0)$$

$$\pi(4) = (1/8 \quad 0 \quad 3/8 \quad 0 \quad 3/8 \quad 0 \quad 1/8)$$

$$\pi(5) = (1/8 \quad 3/16 \quad 0 \quad 3/8 \quad 0 \quad 3/16 \quad 1/8)$$

$$\pi(6) = (7/32 \quad 0 \quad 9/32 \quad 0 \quad 9/32 \quad 0 \quad 7/32)$$

The first and the last entries of $\pi(n)$ are identical and correspond to the cumulative probabilities until n-steps.

We define,

$$\varphi_1(n) = \text{Prob}[\text{first passage through } E_{-1} \text{ at the } n\text{-th interval}]$$

It is to be noted that

$$\varphi_1(2m) = 0, \quad \text{if } L/2 \text{ is even}$$

$$\varphi_1(2m+1) = 0, \quad \text{if } L/2 \text{ is odd}$$

Further,

$$\varphi_1(i) = \pi_{-1}(i) - \pi_{-1}(i-2).$$

5.4.1 MEAN ACCESS TIME

There is a single source of equally likely requests for insertion and retrieval. Generation of requests is stopped if a request waits in a state of buffer overflow or underflow. In order to achieve the specified arrival rate of requests (γ), the source should generate request with an arrival rate γ_e , such that $\gamma = (1 - P_{of} - P_{uf})\gamma_e$. P_{of} and P_{uf} are the probabilities of buffer-overflow and underflow respectively. For a buffer of length L , $P_{of} = P_L$ and $P_{uf} = P_0$. $P_k = \text{Prob}[\text{finding the buffer with } k \text{ items}]$.

In order to determine P_k , we consider an M/M/1/L queueing system in which the system can hold at most a total of L items. Any further arriving item will be refused entry to the system and will depart immediately without service. Requests continue to be generated. For

insertion-rate of λ and retrieval rate of μ , we obtain

$$P_k = P_0 (\lambda / \mu)^k$$

Of course, we also have

$$P_k = 0 \quad \text{for } k > L.$$

Analysis of the finite storage M/M/1/L queueing system is given by L.Kleinrock [1975, Queueing Systems, Vol.1, pp. 103-105]. In order to solve for P_0 we use probability conservation constraint to obtain

$$\begin{aligned} P_0 &= [1 + \sum_{k=1}^L (\lambda / \mu)^k]^{-1} \\ &= \frac{1 - (\lambda / \mu)}{1 - (\lambda / \mu)^{L+1}} \end{aligned}$$

$$\text{For } \lambda = \mu ; \quad P_0 = \frac{1}{L+1}$$

$$\text{and} \quad P_k = P_0 \quad \text{for } 0 \leq k \leq L.$$

Hence we obtain the effective request rate γ_e as follows:

$$\gamma_e = \frac{\gamma}{(1 - \frac{2}{L+1})}.$$

Probability distribution function of the time for the n-th request to occur is computed as below,

$$H_n(t) = \int_0^t \gamma_e \frac{(\gamma_e y)^{n-1}}{(n-1)!} \exp(-\gamma_e y) dy$$

We obtain the mean waiting time for the n-th request,

$$\begin{aligned}
\bar{W}_n &= \int_0^{NT_s} (NT_s - t) d H_n(t) \\
&= NT_s \left[1 - \sum_{j=0}^{n-1} \frac{(\gamma_e NT_s)^j}{j!} \exp(-\gamma_e NT_s) \right] \\
&\quad - \frac{n}{2\gamma_e} \left[1 - \sum_{j=0}^n \frac{(\gamma_e NT_s)^j}{j!} \exp(-\gamma_e NT_s) \right]
\end{aligned}$$

Mean access time, T_b , in a buffered stack memory organization can be computed as follows:

$$T_b = 2 \sum_{n=(L/2)+1}^{\infty} \frac{1}{n} \bar{W}_n \phi_1(n)$$

Mean access time in a stack memory consisting of entirely unidirectional shift-registers is denoted by T_u , that is $NT_s/2$.

5.4.2 PERFORMANCE-IMPROVEMENT (η)

As in Chapter 3, we define the performance-improvement (η) of a buffered stack memory (Figure 3.1(b)) over an entirely unidirectional shift-register stack memory (Figure 3.1(c)) as follows:

$$\eta = \left(1 - \frac{T_b}{T_u} \right) \times 100 \text{ percent}$$

The design objective is to obtain the length of the buffer such that an improvement in performance of the buffered stack approaches that of the ideal stack memory (Figure 3.1(a)), that is 100 percent, and the price/bit that of the unbuffered SR-stack memory (Figure 3.1(c)).

TABLE 5.1 Performance-improvement of a buffered stack
memory with constant clock ECM.
Analytical results.

N = 512

L	MRT($\times T_s$) \longrightarrow									
	25	50	75	100	150	200	250	300	350	400
2	55.9	58.8	61.6	64.5	70.1	75.2	79.5	83.1	86.0	88.4
4	81.1	84.5	87.8	90.8	94.6	96.8	98.0	98.7	99.2	99.4
6	90.5	93.6	96.0	97.6	99.0	99.6	99.8	99.9	99.9	100.0
8	95.4	97.5	98.8	99.4	99.8	99.9	100.0	100.0	100.0	
12	98.7	99.6	99.9	100.0	100.0	100.0				
16	99.7	100.0	100.0							

N = 1024

2	54.5	55.9	57.4	58.8	61.6	64.5	67.3	70.1	72.7	75.2
4	79.4	81.1	82.8	84.5	87.8	90.7	92.9	94.6	95.9	96.8
6	88.9	90.5	92.0	93.6	96.0	97.6	98.5	99.0	99.4	99.6
8	94.3	95.4	96.5	97.5	98.8	99.4	99.7	99.8	99.9	99.9
12	98.1	98.7	99.3	99.6	99.9	100.0	100.0	100.0	100.0	100.0
16	99.3	99.7	99.9	100.0	100.0					

N = 2048

2	53.8	54.5	55.2	55.9	57.4	58.8	60.2	61.6	63.0	64.5
4	78.6	79.4	80.3	81.1	82.8	84.5	86.2	87.8	89.3	90.7
6	88.1	88.9	89.7	90.5	92.0	93.6	94.9	96.0	96.9	97.6
8	93.7	94.3	94.8	95.4	96.5	97.5	98.2	98.8	99.1	99.4
12	97.8	98.1	98.4	98.7	99.3	99.6	99.8	99.9	99.9	100.0
16	99.1	99.3	99.5	99.7	99.9	100.0	100.0	100.0	100.0	

Table 5.1 records the analytical values of η for $N = 512, 1024$ and 2048 ; $L = 2, 4, 6, 8, 12$ and 16 ; and $MRT/T_s = 25, 50, 75, 100, 150, 200, 250, 300, 350$ and 400 . MRT is the mean inter-request time, that is $1/(\lambda + \mu)$.

5.5 VERIFICATION OF THE ABOVE MODEL

In order to verify the analytical results, a simulation study of the proposed scheme was carried out. The program was written in GPSS-III. Inter-request-arrival times are considered to be exponentially distributed. Mean arrival rates for insertion and retrieval requests are assumed to be equal. A request is serviced by the buffer. A request for insertion waits if the buffer is found full at the instant of its arrival. Similarly a request for retrieval waits if the buffer is found empty at the instant of its arrival. The buffer is initialised to half of its capacity after a fixed period of NT_s .

A simulation experiment runs for 5000 requests. Confidence intervals of the mean access time and the performance-improvement of the buffered stack memory, are determined as follows. Let σ be the standard deviation of the waiting time. Standard deviation for the mean waiting time, T_b , can be estimated as σ/\sqrt{S} , where S is the number of requests in a simulation experiment. The 95 percent confidence-interval for mean access time lies between $(T_b - 1.96\frac{\sigma}{\sqrt{S}})$ and $T_b + 1.96\frac{\sigma}{\sqrt{S}}$, and that for the

performance-improvement lies between $(\eta - c)$ and $(\eta + c)$

where $c = \frac{2 \times 1.96 \sigma}{N \sqrt{S}} \times 100$.

Tables 5.2(a) and (b) compare the analytical results with the simulation results for $N = 512$ and 1024 respectively. Each table records the values for T_b and η for $L = 2, 4, 6, 8$ and 12 ; and $MRT/T_s = 25, 50, 75, 100, 150, 200, 300$ and 400 .

In many applications where we want to fit a known mathematical distribution, the gamma distribution gives a reasonable fit. Erlang distribution is a specific case of the gamma distribution with an integer Erlang constant E . What Erlang conceived was the notion of decomposing the service time distribution into a collection of structured exponential distributions [L. Kleinrock, 1975]. Table 5.3 records the simulation values of T_b and η for $MRT/T_s = 100, 200, 300$ and 400 ; $L = 2, 4, 8$ and 12 ; and $E = 1, 2$ and 8 . $E = 1$ corresponds to an exponential distribution and $E = \infty$ to a constant distribution. It is observed from Table 5.3 that η increases and c decreases with an increase in E . Thus the increase in E results in the decrease of both the mean access time and the variance. These are the expected results.

So far we considered a request for a single insertion or a single retrieval. In many applications, a request occurs in the form of a burst. Burst-length may

TABLE 5.2 Buffered stack memory with constant clock ECM.
Comparison of analytical and simulation results.

(a) $N = 512$, $E = 1$.

SIMULATION						ANALYTICAL					
SIMULATION						ANALYTICAL					
MRT ($x T_s$)	L	T_b	$\eta \pm c$	T_b	η	MRT ($x T_s$)	L	T_b	$\eta \pm c$	T_b	η
25	2	123.6	52.0 \pm 2.2	112.8	55.9	50	2	108.4	97.9 \pm 2.0	105.5	98.8
	4	40.0	84.4 \pm 1.3	48.3	81.1		4	31.3	87.9 \pm 1.1	39.7	84.5
	6	17.6	93.4 \pm 0.8	24.4	90.5		6	11.5	95.8 \pm 0.6	16.5	93.6
	8	8.6	96.9 \pm 0.6	11.8	95.4		8	4.7	98.5 \pm 0.4	6.5	97.5
	12	2.6	99.3 \pm 0.3	3.2	98.7		12	0.2	100. \pm 0.1	0.9	99.6
75	2	98.7	61.8 \pm 1.8	98.2	61.6	100	2	92.4	63.9 \pm 1.9	90.9	64.5
	4	26.3	89.9 \pm 1.0	31.2	87.8		4	20.8	92.2 \pm 0.8	23.9	90.7
	6	6.1	97.5 \pm 0.6	10.2	96.0		6	5.6	98.1 \pm 0.4	6.2	97.6
	8	2.7	99.3 \pm 0.3	3.1	98.8		8	1.7	99.3 \pm 0.4	1.5	99.4
	12	0.2	100. \pm 0.1	0.3	99.9		12	0.1	100. \pm 0.1	0.1	100.
150	2	73.6	70.9 \pm 1.9	76.6	70.1	200	2	71.8	72.3 \pm 1.7	63.6	75.2
	4	16.5	93.8 \pm 0.7	13.7	94.6		4	12.4	95.4 \pm 0.6	8.1	96.8
	6	3.1	98.9 \pm 0.3	3.5	99.0		6	2.5	99.3 \pm 0.6	1.1	99.6
	8	0.4	99.8 \pm 0.1	0.4	99.8		8	0.4	100. \pm 0.1	0.1	99.9
	12	0.0	100.	0.0	100.		12	0.0	100.	0.0	100.
300	2	53.0	79.3 \pm 1.3	43.2	83.1	400	2	44.0	82.8 \pm 1.2	29.8	88.4
	4	6.6	97.7 \pm 0.4	3.2	98.7		4	4.6	98.3 \pm 0.4	1.5	99.4
	6	1.1	99.7 \pm 0.2	0.3	99.9		6	0.2	100. \pm 0.1	0.1	100.
	8	0.1	100. \pm 0.1	0.0	100.		8	0.1	100. \pm 0.1	0.0	100.
	12	0.0	100.	0.0	100.		12	0.0	100.	0.0	100.

TABLE 5.2 Buffered stack memory with constant clock ECM.
Comparison of analytical and simulation results.

(b) $N = 1024$, $E = 1$.

RT ($\times T_s$)	SIMULATION					MRT ($\times T_s$)	ANALYTICAL				
	L	T_b	$\eta \pm c$	T_b	η		L	T_b	$\eta \pm c$	T_b	η
25	2	286.2	44.2 \pm 2.4	232.9	54.5	50	2	243.6	52.6 \pm 2.2	225.6	55.9
	4	94.4	81.7 \pm 1.5	105.3	79.4		4	79.9	84.6 \pm 1.3	96.6	81.1
	6	40.4	92.2 \pm 1.8	57.0	88.9		6	36.5	93.0 \pm 1.7	48.9	90.5
	8	21.9	95.9 \pm 1.3	29.2	94.3		8	17.6	96.7 \pm 0.6	23.6	95.4
	12	8.0	98.5 \pm 0.8	9.7	98.1		12	5.3	99.1 \pm 0.3	5.5	98.7
	16	3.5	99.5 \pm 0.5	3.1	99.4		16	1.3	99.9 \pm 0.1	1.4	99.7
75	2	233.6	54.5 \pm 2.2	218.3	97.4	100	2	220.0	57.3 \pm 2.0	211.0	58.8
	4	69.2	86.6 \pm 1.2	88.0	82.8		4	60.5	88.3 \pm 1.1	79.3	84.5
	6	29.9	94.4 \pm 1.4	40.8	92.0		6	24.0	95.6 \pm 1.2	33.0	93.6
	8	11.5	97.7 \pm 0.8	18.0	96.5		8	8.7	98.5 \pm 0.4	12.9	97.5
	12	3.2	99.5 \pm 0.4	3.6	99.3		12	1.1	99.9 \pm 0.1	1.9	99.6
	16	0.4	100. \pm 0.1	0.4	99.9		16	0.0	100.	0.1	100.
50	2	193.0	62.3 \pm 1.8	196.4	61.6	200	2	185.0	65.9 \pm 1.9	181.8	64.5
	4	51.5	90.1 \pm 1.0	62.5	87.8		4	45.8	91.3 \pm 0.8	47.8	90.7
	6	16.8	95.4 \pm 1.0	20.4	96.0		6	12.8	97.7 \pm 0.8	12.5	97.6
	8	5.3	99.1 \pm 0.3	6.3	98.8		8	3.3	99.4 \pm 0.3	3.1	99.4
	12	0.6	99.9 \pm 0.1	0.5	99.9		12	0.4	100. \pm 0.1	0.2	100.
00	2	150.0	70.7 \pm 1.9	153.3	70.1	400	2	147.0	70.9 \pm 1.7	127.1	75.2
	4	28.2	94.5 \pm 0.7	27.5	94.6		4	26.7	94.7 \pm 0.6	16.2	96.8
	6	8.1	98.5 \pm 0.6	5.0	99.0		6	4.4	99.3 \pm 0.4	3.1	99.4
	8	1.3	99.5 \pm 0.1	0.9	99.8		8	0.6	99.9 \pm 0.1	0.3	99.9
	12	0.2	100. \pm 0.1	0.0	100.		12	0.0	100.	0.0	100.

TABLE 5.3 Buffered stack memory with constant clock ECM.
Effect of variation in Erlang distribution on
performance. Simulation results. $N = 512$

	E=1			E=2		E=8	
MRT (x T _s)	L	T _b	η±c	T _b	η±c	T _b	η±c
100	2	92.4	63.9±1.9	90.6	64.9±1.7	73.6	71.6±1.7
	4	20.8	92.2±0.8	19.0	92.6±0.8	19.0	92.6±0.7
	8	2.7	99.3±0.3	1.5	99.7±0.2	1.0	99.7±0.1
	12	0.2	100.0±0.3	0.1	100.0±0.1	0.0	100.0
200	2	71.8	72.3±1.7	60.8	73.6±1.3	60.0	76.8±1.3
	4	12.4	95.4±0.6	8.9	96.9±0.5	5.8	98.1±0.3
	8	0.4	100.0±0.1	0.1	100.0	0.0	100.0
	12	0.0	100.0	0.0	100.0	0.0	100.0
300	2	53.0	79.3±1.3	46.2	81.9±1.1	42.0	83.6±1.0
	4	6.6	97.7±0.4	3.9	98.9±0.3	0.8	100.0±0.1
	8	0.3	100.0±0.1	0.0	100.0	0.0	100.0
	12	0.0	100.0	0.0	100.0	0.0	100.0
400	2	44.0	82.8±1.2	34.4	87.3±0.9	22.4	91.2±0.6
	4	4.6	98.3±0.4	1.7	99.3±0.2	0.1	100.0
	8	0.1	100.0	0.0	100.0	0.0	100.0
	12	0.0	100.0	0.0	100.0	0.0	100.0

MRT = Mean inter-request time, L = Length of buffers
(fast stack)

T_b = Mean access time, η = Performance improvement.

c = Confidence interval.

TABLE 5.4 Effect of burst-length on the performance of a buffered stack memory with constant clock ECMS. Simulation results. $N = 512$. $MRT = 100T_s$.

MRT=100T _s				MRT = 100T _s			
\bar{b}	L	$T_{b \pm c}$ (x T _s)	T_b/NT_s	\bar{b}	L	$T_{b \pm c}$ (x T _s)	T_b/NT_s
1	2	90.8±4.5	0.177	3.0	2	706.8±18.2	1.40
	4	21.0±2.1	0.041		4	305.0±9.4	0.595
	8	1.4±0.4	0.003		8	95.3±4.8	0.186
	12	0.1±0.1	0.000		12	40.0±2.8	0.080
	16	0.0	0.000		16	19.4±0.1	0.038
1.5	2	228.4±8.2	0.446	4.0	2	1049.4±28.5	2.05
	4	71.6±4.0	0.140		4	460.7±12.1	0.900
	8	12.7±1.6	0.025		8	171.5±4.3	0.335
	12	2.8±0.7	0.005		12	74.1±4.0	0.145
	16	0.1±0.1	0.000		16	37.4±2.9	0.073
2.0	2	420.0±12.0	0.820	4.5	2	1123.5±32.6	2.19
	4	144.1±5.9	0.281		4	539.0±15.2	1.03
	8	32.4±2.7	0.063		8	212.6±7.5	0.410
	12	11.6±1.4	0.022		12	98.4±5.0	0.192
	16	3.7±0.7	0.007		16	48.0±3.3	0.093
2.5	2	564.5±16.0	1.100	\bar{b} = Mean burst-length MRT = Mean inter-request time T_b = Mean access time $T_{b \pm c}$ = 95 percent confidence-limits for T_b .			
	4	216.7±7.8	0.423				
	8	56.0±3.5	0.109				
	12	22.0±2.2	0.042				
	16	9.4±1.4	0.018				

follow a certain distribution. We conducted a simulation study with uniformly distributed burst-lengths. A burst-request waits if buffer becomes either full or empty during the recirculation period. Probabilities of buffer-underflow and overflow are expected to increase with an increase in burst-interval. The ratio of the mean access time to the recirculation period, NT_s is considered a figure of merit. Table 5.4 records T_b and T_b/NT_s for $L = 2, 4, 8, 12$ and 16 ; $MRT = 100 T_s$; $N = 512$; and mean burst-length, $\bar{b} = 1, 1.5, 2., 2.5, 3., 4.$ and 4.5 .

5.6 DISCUSSION OF RESULTS

Table 5.1 serves as a design guide table. To illustrate with an example, for $L = 4$, and $MRT = 100 T_s$, $\eta = 90.7$ percent for $N = 512$ and $\eta = 84.5$ percent for $N = 1024$. Tables 5.2(a) and (b) compare the analytical results with the simulation results for $N = 512$ and 1024 . It is to be noted that η depends on the product $\rho = \gamma NT_s$, where $1/\gamma$ is the mean inter-request time (MRT). ρ denotes the average number of requests occurring during NT_s . We define a constant f equal to $10.24.\rho$ can be expressed in terms of f as given in Table 5.5.

It can be verified from Tables 5.1 and 5.2 that η for a specified value of ρ is same for $N = 512, 1024$ and so on. The simulation values tally within the specified confidence interval.

TABLE 5.5
 ρ in terms of f .

N	MRT ($\times T_s$)							
	25	50	75	100	150	200	300	400
512	$2f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$	$\frac{1}{3}f$	$\frac{1}{4}f$	$\frac{1}{6}f$	$\frac{1}{8}f$
1024	$4f$	$2f$	$\frac{4}{3}f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$	$\frac{1}{3}f$	$\frac{1}{4}f$
2048	$8f$	$4f$	$\frac{8}{3}f$	$2f$	$\frac{4}{3}f$	f	$\frac{2}{3}f$	$\frac{1}{2}f$

For η greater than 95 percent with $N = 512$ a buffer of length $L = 6$ is selected for $\rho \leq f/2$, that is $\text{MRT} \geq 100 T_s$.

The plots of η versus buffer-length are shown in Figure 5.4 for $\rho = 2f, f/2$ and $f/4$. The analytical curves are fairly close to the corresponding simulation curves. For $\rho > f/2$, analytical values of η are slightly less than the simulation values and hence yield a conservative estimate for L . For $\rho < f/2$ analytical values of η are slightly more than the corresponding simulation values.

The plots of η versus MRT are shown in Figure 5.5 for $L = 2, 4$ and 6 . The analytical curves are close to the simulation curves except for lower values of L with $\rho < f/5$ and $\rho > 2f$.

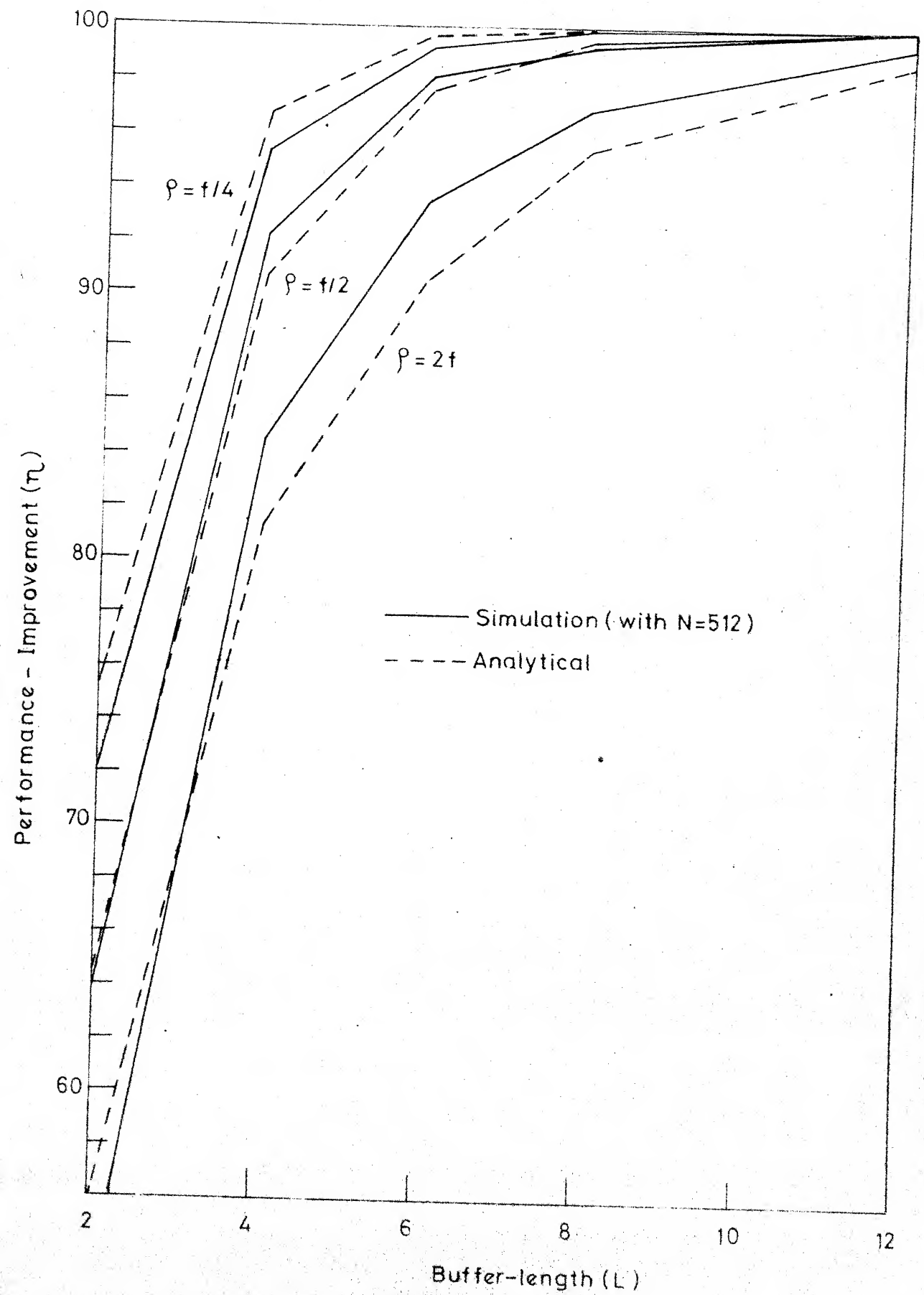


Figure 5.4 BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM.

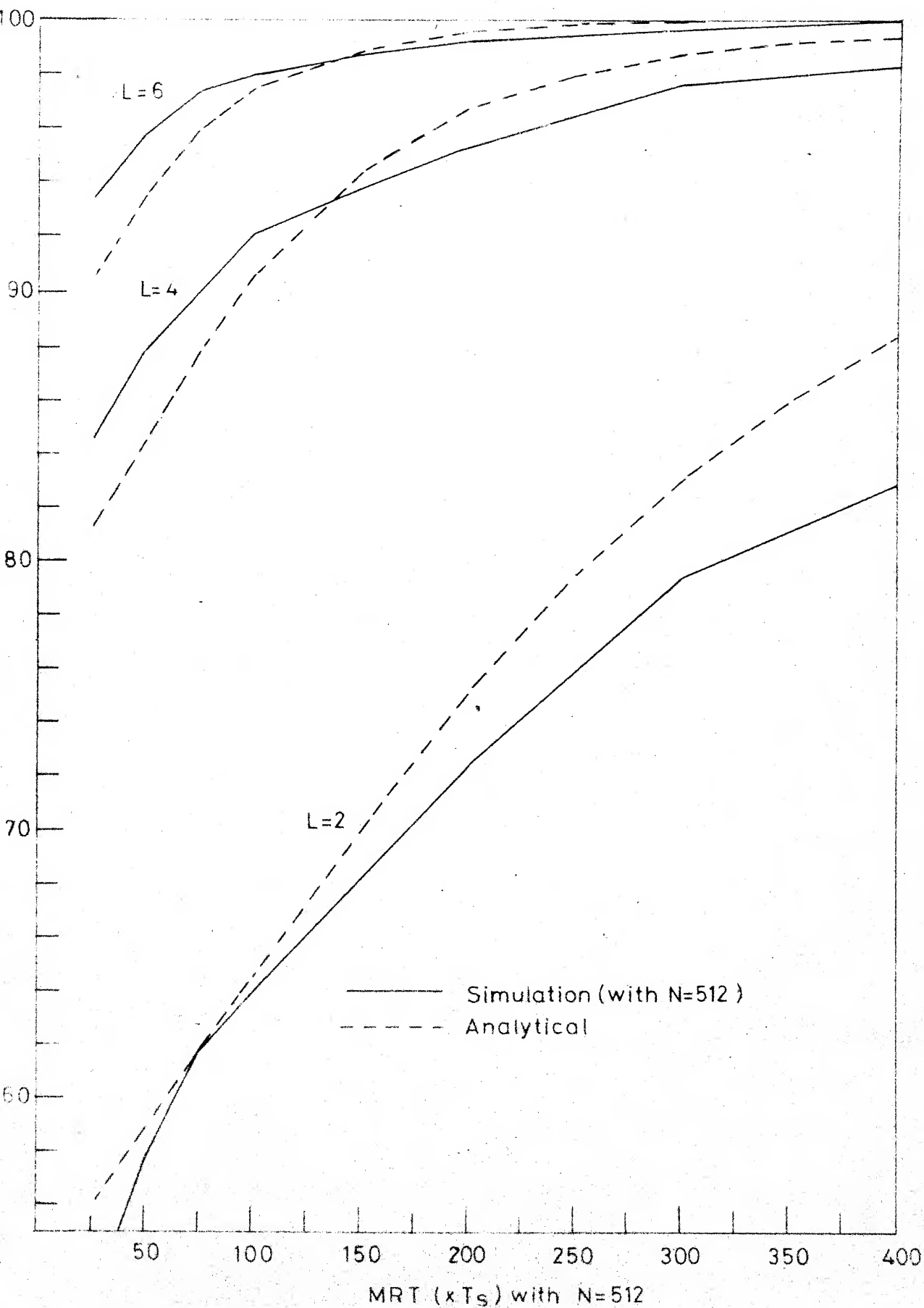


Figure 5.5 BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM.

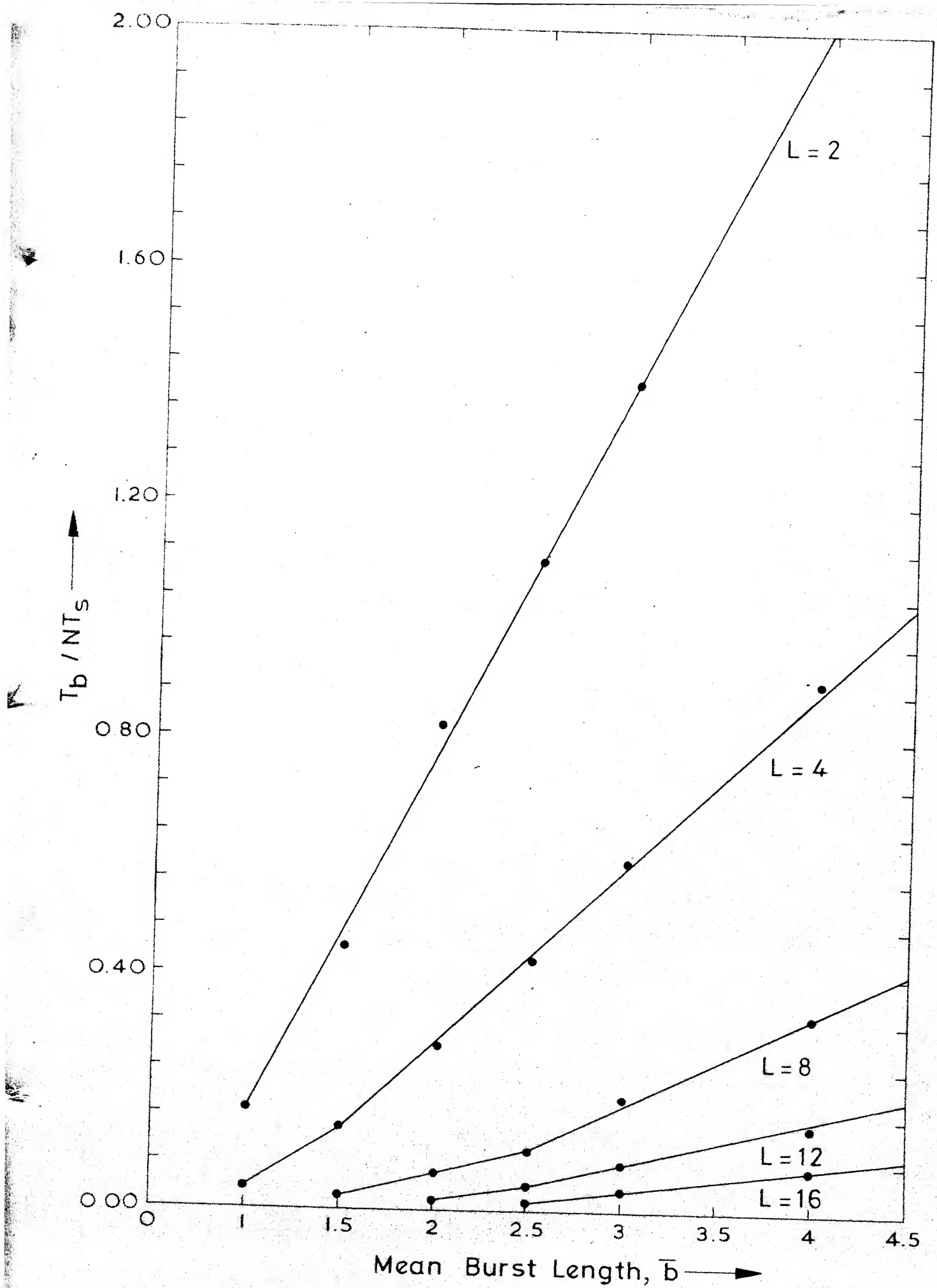


Figure 5.6 EFFECT OF BURST-LENGTH ON PERFORMANCE OF A BUFFERED STACK WITH CONSTANT CLOCK ECM.

Effect of Erlangian distribution on the performance of the buffered stack memory can be observed from Table 5.3. The simulation-study of the buffered stack memory verifies that the mean access time and its variance decrease with an increase in the value of Erlang constant, E .

The plots of T_b/NT_s versus mean burst length (\bar{b}) are shown in Figure 5.6 for $L = 2, 4, 8, 12$ and 16 ; and $N = 512$. We observe that the mean access time increases linearly at the rate of $0.61 NT_s$, $0.30 NT_s$ and $0.15 NT_s$ per unit of mean burst-length for $L = 2, 4$ and 8 beyond the mean burst length, $\bar{b} = 1, 1.5$ and 2.5 respectively.

5.7 APPLICATIONS

Also refer to Chapter 3. A stack facilities recursion enables fast interrupt handling, increases speed of computation by virtue of code compression which is inherited from its implicit addressing, provides one for one correspondence between source language operators and machine instructions, preserves code and data conditions to facilitate rapid environment change [E.I. Organick, 1973, R.W.Doran, 1975].

Application of stacks can be noticed in multi-programming, subroutine operations, language translation, dynamic programming, efficient processing of object programs written in structural language, like ALGOL.

MU5, HP3000, KDF9, B5500, B6700, B7700, and PDP-11 use stacks. In order to reduce storage references, these computers use top-of-stack registers. The HP3000 has four such registers and B7700 has as many as 32 [R.W. Doran, 1975].

5.8 CONCLUSION

A scheme has been proposed to design a buffered stack memory with constant clock ECMs. Addition of small size high speed registers (FS) on top of a large size stack with ECMs reduces the mean access time drastically. Buffered organization overcomes the asymmetric access to shift-register stack by virtue of periodic initialization of the fast stack (FS) to half of its capacity. Improvement in performance over an unbuffered shift-register stack is considered the design criterion.

The design guide table is prepared to facilitate the selection of a buffer-length for the given values of the mean inter-request time (MRT), the performance-improvement (η) and the size of shift-register stack (N). For instance, a fast stack of size 6 may be selected to achieve the performance-improvement (η) greater than 95 percent with a shift-register stack of size $N = 512$ for $MRT \geq 100 T_s$.

CHAPTER 6

ASSOCIATIVE ECM ORGANIZATION

In this chapter a conceptual design of an associative electronic cyclic memory is proposed. An associative operation is performed in two phases. Search operation is carried out in phase 1, and follow-up operations in phase 2. m large size shift-registers together with m small size buffers form a basic cyclic memory module (BCM) which provides m -bits of a word in parallel and words are sequentially accessible (see Figure 6.1) size of the content addressable memory (CAM) is about one fourth of the size of the array of BCMs. Arithmetic and logical operations are performed by a processing unit (PU). Addition of parallel-in-serial-out (PISO) shift-registers enables one to perform string manipulative operations. Multiple search operations over a word during a single shift-period enhances the parallelism.

6.1 INTRODUCTION

Associative operations are based on content addressing rather than location addressing. The basic associative operation is SEARCH for a record with a key satisfying a specified relation with a given argument. Relations may be 'equal to', 'less than', 'between limits', 'next higher', 'maximum', and their counterparts.

Follow-up operations may be 'pick first match', 'pick all matches', 'find its address', 'read a record with successful match', 'simple or multi-write in the specified location and so on. A sequence of arithmetic operations may precede a WRITE operation.

A comparison logic unit is required to perform SEARCH operations. Distribution of comparison logic over memory cells gives rise to a wide spectrum of associative memories (AM) which includes fully parallel, word-parallel and bit-serial, bit-parallel and word-serial and block oriented random access memories. Fully parallel associative memory is most expensive as comparison logic is associated with each bit. Bit-parallel memory is less expensive than word-parallel but results in excessive processing time. Block oriented random access memory (BORAM) appears to be suitable for information storage and retrieval [B.Parhami, 1973].

Slotnick [1970] proposes a logic-per-track disk memory which is suitable for applications which suffer from high cost of random access memories or performance degradation due to frequent transfers between primary and secondary memories. Lee's distributed logic memory [C.Y.Lee,1963] has attracted attention of many researchers. It consists of a control unit and a large number of identical cells, each of which stores one character of information. Cells

can communicate with two adjacent neighbours and the control unit. Cells can evaluate input conditions independently and in parallel. Operations on variable length data become easier in such an organization.

Electronic cyclic memory (ECM) suffers from large access time due to sequential access. However, it has advantage of very high data rates. The mobility of information bits and the ability to vary clock rates facilitate string manipulative operations.

The high ratio of data area to key area in a record enables the designer to reduce the size of CAM. Sequential comparison of a data word stored in CAM with a set of search arguments during a single shift-period will improve the parallelism.

6.2 DESCRIPTION OF THE SCHEME

Associative Electronic Cyclic Memory consists of an array of basic cyclic memory modules (BCMs), match flip-flops, routing logic unit, processing unit (PU), a few parallel-in-serial-out (PISO) shift-registers and a content addressable (bipolar) memory of a few words.

A BCM consists of m storage planes to provide m bits of a word in parallel (refer to Figure 6.1). Each storage plane consists of a small size buffer and a large size ECM with variable clock rate, e.g., static shift-registers (SRs). Let L and N be their respective sizes.

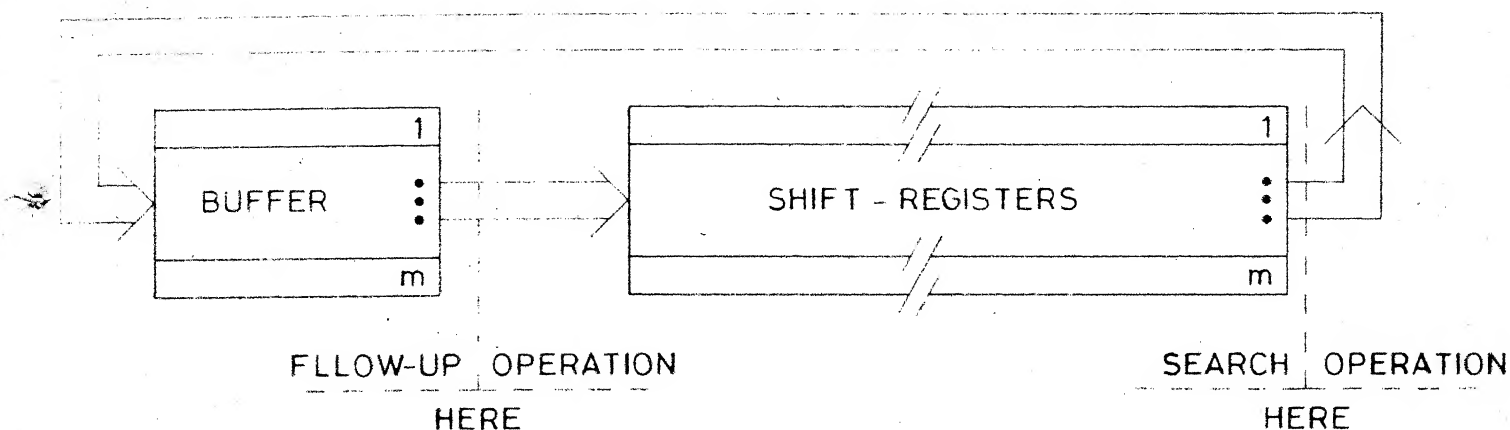


FIG. 6.1 (a) BASIC CYCLIC MEMORY MODULE (BCM)

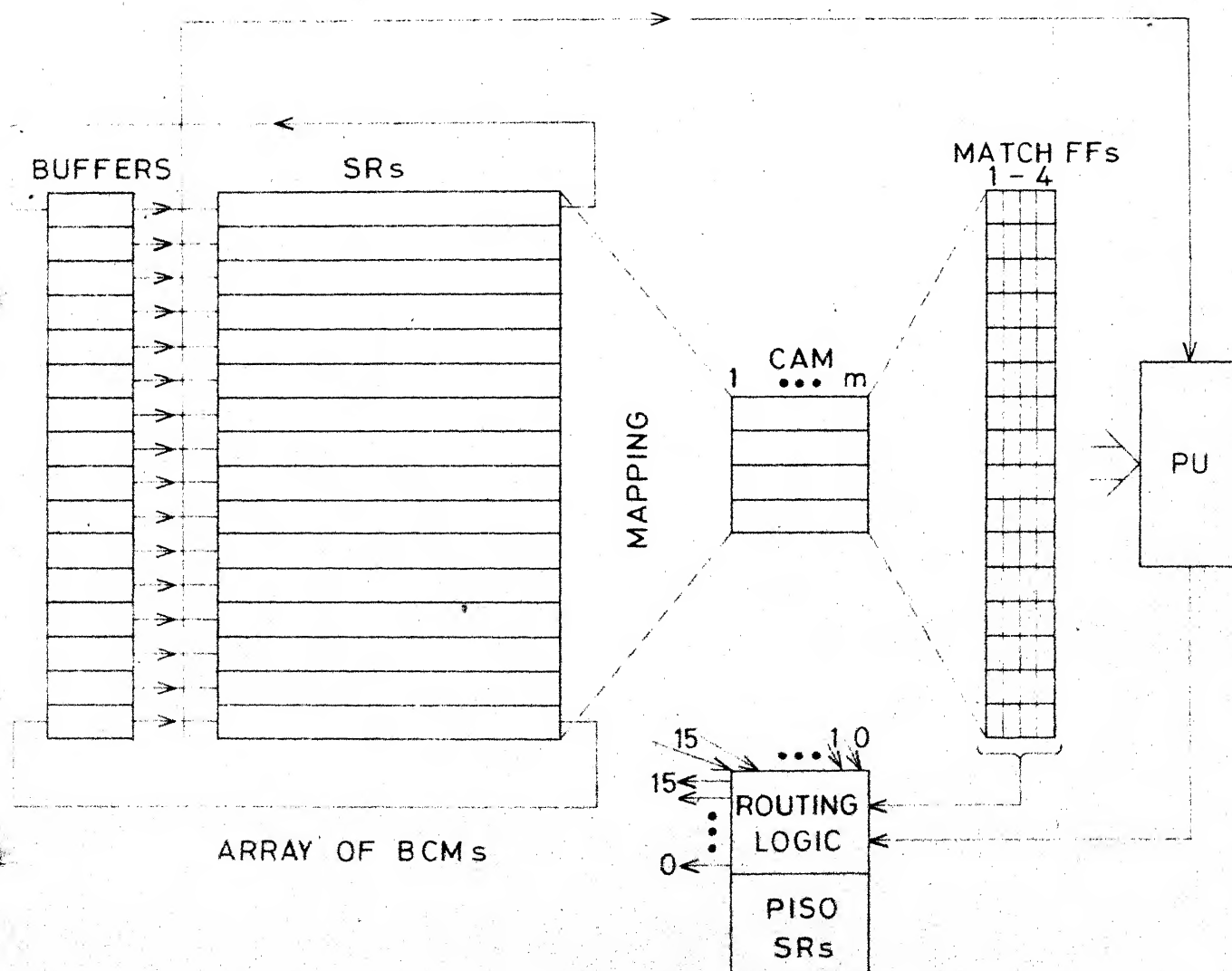


FIG. 6.1 (b) AN ASSOCIATIVE ECM ORGANIZATION

The content addressable memory is of q words where q is about one fourth of the size (a) of array of BCM's. pxq flip-flops store the match response for p -search arguments being compared consecutively during a shift-period. Processing unit (PU) performs arithmetic and logical operations as required after a successful match. Routing logic unit facilitates data movement from one BCM to another. 'Copy file' operation requires only change in data path. PISO shift-registers are useful to perform the operations, like 'insertion of a string', 'deletion of a string' and 'garbage collection'. Clock to a BCM can be independently controlled. It can be inhibited for a certain interval of time.

6.3 OPERATION

Refer to Figure 6.1. Information within a BCM rotates past the output port of SRs. As the beginning of a record is detected at the output port of SRs and CAM is found busy, the corresponding clock is inhibited. SEARCH operation is performed within CAM. At the completion of a search operation over key area, CAM selects the ready BCMs with inhibited clocks and SEARCH operation is initiated. As comparison within a CAM is much faster, multiple comparison operations can be performed during a single shift-period. Match response corresponding to each search operation is stored in match flip-flops associated with those BCMs. Key area words move through the buffer. As the

beginning of a record in a BCM reaches the output end of the buffer, the corresponding match flip-flops are interrogated by the processing unit. Subsequently the clock to that BCM is inhibited if any follow-up operation is to be performed over that record. Otherwise information keeps recirculating with the normal clock-rate. In the case of a change of data path PU transfers control to the routing logic unit and becomes available to execute other operations. Routing is, essentially, connecting the output end of buffer to the input of a specified BCM. PISO shift-registers are used to perform operations, like 'Insertion of a String', 'Deletion of a string' and 'garbage collection' [Om Vikas and V. Rajaraman, 1976].

A counter is associated with every BCM. It is set to zero at the start of search operation. It is incremented with a shift clock. The search operation for a given set of arguments is complete over the entire storage, when all the counters are reset to zero.

6.4 DESIGN METHODOLOGY

A record can be partitioned into two parts, namely key-area and data-area. Search is to be performed over key-area. Let key-area be of fixed length, say, 16 words. A word may be a byte or a character. Data-area is assumed to be of variable length. Let its mean length be 48 words. Hence about 16 records can be stored in a BCM of 1024 words. Clock-rate may vary from 100 KHz to 5 MHz,

that is a shift-period may vary from 200 nsec to 10 μ sec. In other words, a clock can be inhibited upto 50 normal shift-periods. 16 shift-periods are used up in performing a search operation over a key-area. The worst case occurs when the beginning of records coincides for all BCMs. Three groups of BCMs can be interrogated without exceeding the permissible clock delay. Hence the size of CAM should be one third of the size of array of BCMs. However, this event is a rare one and may be overcome by 'inhibit-ahead' technique, that is clocks to some of BCM's are put to normal rate when majority-coincidence event takes place. Majority-coincidence event can be defined as the occurrence of 3q or more BCM's showing the beginning of records. This technique enables one to reduce the size of CAM which is an expensive ingredient. We consider the ratio $q/a = 1/4$.

A comparison operation in a CAM is complete within 40 nsec., whereas a normal shift-period lasts over 200 nsec. This time-differential facilitates four sequential search operations to be performed safely. Hence $p = 4$.

The search response is available at the end of key-area. Follow-up operation is performed depending on the search response. Key-area is stored in the buffer and is available for the desired operations which would have not been possible in the absence of such a buffer. The size of buffer, L , is equal to the length of key area. In our example $L = 16$.

Design of PISO shift-registers being used for insertion/deletion of a string is discussed in Chapter 4 and also in [Om Vikas and V. Rajaraman, 1976].

The processing unit can be a microprocessor.

6.5 APPLICATIONS

Figure 6.1 is a conceptual schematic diagram of an associative ECM organization being used for non-numeric applications, e.g., symbol manipulation, and information storage and retrieval.

6.6 CONCLUSION

A conceptual design of an associative electronic cyclic memory is presented. This seems to be a cost-effective memory organization for microprocessor-based information storage and retrieval systems.

The proposed scheme is suitable for variable length records. Search and follow-up operations are performed at different points which enhances the performance and simplifies the logic circuitry. Insertion and deletion operations are performed during the same cycle in which the search succeeds. Garbage collection operation can be performed as a special deletion operation.

CHAPTER 7

CONCLUSIONS AND TOPICS FOR FURTHER RESEARCH

In this final chapter, we present the significant results of the thesis and attempt to put them in some perspective with respect to their potential for practical applications. The research topics which are deemed to be of immediate importance are also mentioned.

7.1 SUMMARY OF RESULTS

New cyclic memories, like CCDs and Magnetic Bubbles fill the price-performance gap between semiconductor/core RAM, and magnetic disk and drum. We put these memories in the class of Electronic Cyclic Memory (ECM). Two characteristics, namely the mobility of information and the ability to vary clock rate make them suitable for non-numeric processing, stack-based operations and microprocessor-based application systems.

We propose buffered ECM organizations with various access disciplines, namely FIFO and LIFO. We observe that the appropriate buffering of an ECM considerably enhances its performance. In particular, the periodic initialization of buffers with constant clock ECMs could yield even 100 percent improvement.

A scheme for a buffered ECM with FIFO access discipline is discussed in Chapter 2. Two small buffers

on both of the ends of an ECM are used to cater to random input-output requirements. It is concluded that it is preferable to use a constant clock-rate. Table 2.1 serves as the design-guide table with performance-improvement as a design criterion. Buffers of size 6 added to an ECM of size 512 exhibit an improvement in performance by more than 95 percent for a mean inter-request-time, $MRT \geq 100 T_s$. In order to compute the fractional loss of information in a buffered serial memory, a computationally convenient analytic method is developed for the analysis of M/G/1/L and G/M/1/L finite queue Skinner's models. These two models are shown to be duals. Table 2.3 records the fractional loss of information and can be used to select an optimal buffer-length. For example, buffers of size 12 with a constant clock ECM of size 512 yield a fractional loss of information of less than 10^{-3} . Simulation study was carried out to verify the above models. For the loss system, the analytical results agree well with the simulation results. However, in the wait system, the analytical values differ slightly with the corresponding simulation values for buffer lengths which are much larger or much smaller than $\frac{1}{2} \gamma NT_s$, where γ is the mean request-rate. Figure 2.6 shows a fast increase in the value of the performance-improvement (η) with a small increase in the buffer-length.

We have proposed two buffered stack memory organizations with variable clock ECM in Chapter 3, and with constant clock ECM in Chapter 5. A queueing model of the buffered stack memory with variable clock ECM is developed from first principles. Table 3.1 records the analytical results and can be used as a design-guide table with the performance-improvement as the design-criterion. The analytical values agree well with the simulation values except for very small values of buffer-length.

The design of a buffered stack memory with constant clock ECM employs the novel technique for multiple insertions and deletions in an ECM which is discussed in Chapter 4. This technique facilitates multiple insertions and deletions by means of expanding and contracting the data path.

Table 5.1 records the analytical values of performance-improvement and can be used as a design-guide table. The analytical results agree well with the simulation results except for the buffer lengths which are much larger or much smaller than $\frac{1}{2} \gamma NT_s$. A comparison of the buffered stack memory with variable and constant clock ECM is shown in Figure 7.1. Variable clock ECM exhibits better performance for buffer length $L = 2$. For $L \geq 4$, the performance improves faster with constant clock ECM. The performance-improvement of 100 percent could be achieved with constant clock ECM of size 512 by using a buffer of size 8-16.

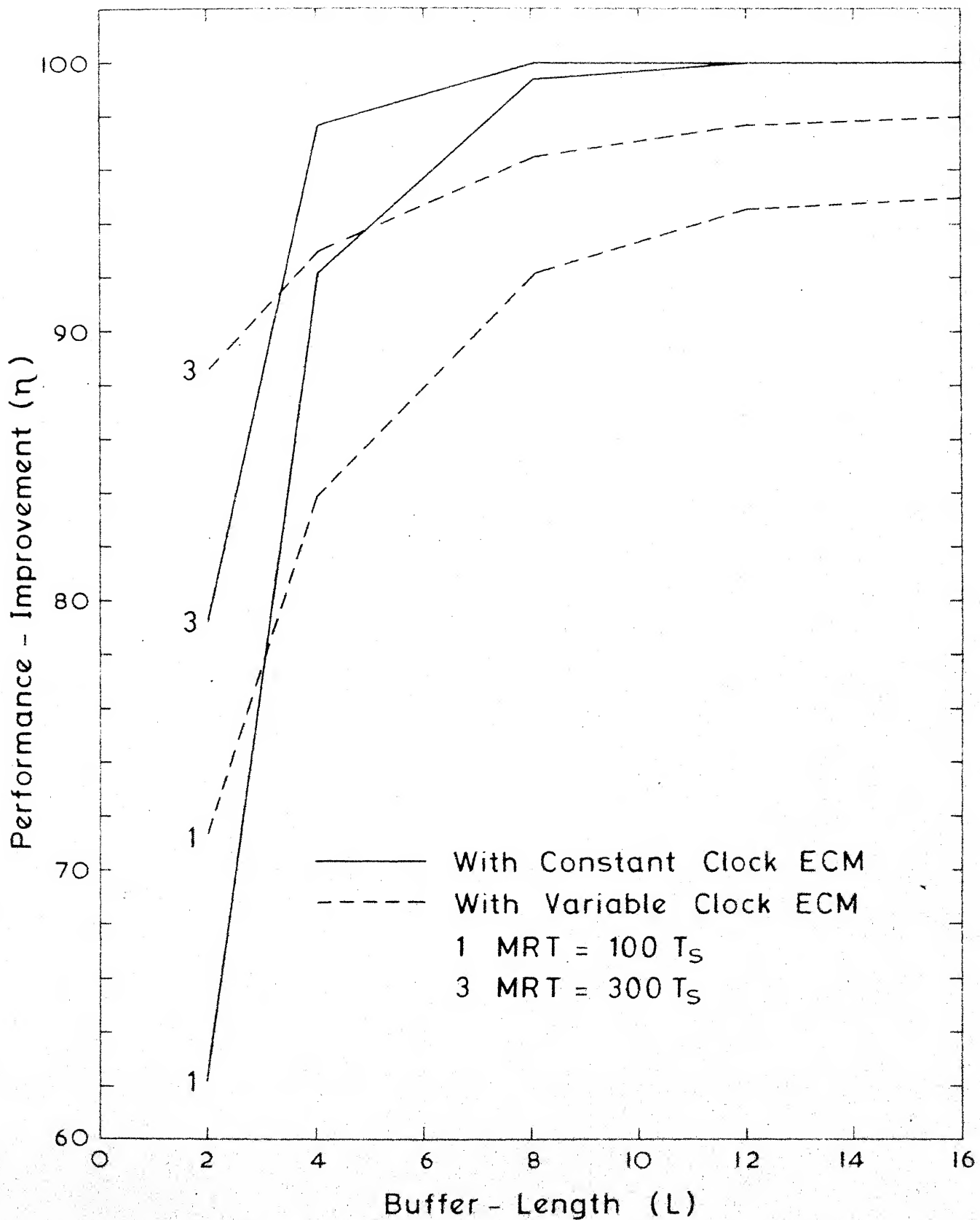


Figure 7.1 PERFORMANCE COMPARISON OF BUFFERED STACK MEMORY WITH VARIABLE AND CONSTANT CLOCK ECMs. SIMULATION RESULTS WITH $N = 512$.

Simulation study of the above schemes shows reduction in the mean and the variance of access time (refer to Tables 3.3 and 5.3).

Effect of variable length burst-insertions and retrievals is also studied (refer to Figures 4.4 and 5.6). Mean access time increases linearly with an increase in burst-length, ℓ for $\ell > \bar{b}$, where $\bar{b} = (L+2)/4$. It is desirable that an insertion or deletion operation be complete within at most two cycles. Hence the buffer-length is selected such that $1+2(\bar{b}-1) \leq \frac{L}{2}$. \bar{b} is the mean burst-length.

An associative ECM organization is proposed in Chapter 6. The ability to vary clock-rate and the high ratio of data area to key area are exploited to reduce the size of the content addressable memory which is an expensive ingredient in this system.

Simulation is a very expensive design tool. We have given analytic methods to determine buffer-length. These methods are easy to use in practice.

7.2 TOPICS FOR FURTHER RESEARCH

Emergence of electronic cyclic memories has a major impact on computer architecture. The ability to vary clock rate and the property of information-mobility in such memories open a new horizon for their application in various areas, namely microprocessor based systems, non-numeric processing,

and stack-oriented applications. Hence there is need to pursue research in this area.

The following is the list of topics which need further research.

Hardware

- * Development of a micro-controller for an ECM to be used in a computer system is of paramount importance. Various scheduling disciplines and data organizations can be considered.

- * A conceptual design of an ~~mass~~associative ECM organization is discussed in the thesis. Detailed logical design and fabrication still remain to be done.

Data Structure

- * We feel that ECM will emerge as cost-compatible memory for micro-processor based systems for non-numeric processing. The problem of data structure in an ECM requires immediate attention. Due to the property of information-mobility and varying clock-rate, the emergence of more efficient data structures is postulated.

Analysis

- * Analysis of a buffered electronic cyclic memory with other scheduling disciplines, like Shortest Latency Time First (SLTF) and Shortest Processing Time First (SPTF), and data organizations, like file and page will

be useful in performance-evaluation. Fuller's analysis of rotating storage units [1975] may be the guiding material in the initial stages of the research work.

- * Analytic models are needed be developed for bulk insertions and retrievals in FIFO and LIFO access disciplines.

- * Analysis of the proposed scheme for multiple insertions and deletions in a shift-register will be useful to quantify the performance of microprocessor-based systems where this scheme will be widely applicable.

- * We have suggested in this thesis a computationally convenient method to analyse the finite queue Skinner's model with server-walking time. This is with FIFO access discipline. Development of a similar model with LIFO access discipline will be useful to study the behaviour of buffered stacks.

Micro-computer Architecture

- * Kartashev [1976] proposes a design of a micro-computer using a shift-register memory. Buffered SR stack-based design of a micro-computer will be an interesting project.

LIST OF REFERENCES

- [1] Abraham, P.W., 'Symbol Manipulation Languages', Advances in COMPUTERS, Vol.9, Academic Press, N.Y., 1968, pp. 51-110.
- [2] Asai, H. and Lee, S.C., 'Design of Queueing Buffer Register Size', Information Processing Letters, pp. 147-152, May 1975.
- [3] Barton, R.S., 'Ideas for Computer System Organization, a personal survey', Software Engineering (Coins III), edited by Tou, J.T., Academic Press, N.Y., Vol.1, pp. 7-16, 1970.
- [4] Beausoleil, W.F., et.al., 'Magnetic Bubble Memory Organization', IBM Journal Res. Develop., Vol.16, pp. 587-591, Nov. 1972.
- [5] Bhandarkar, D.P. and Juliussen, J.E., 'Computer System Advantages of Magnetic Bubble Memories', Computer, pp. 35-40, Nov. 1975.
- [6] Bhat, U.N., 'Some Problems in Finite Queues', Conf. Michigan Univ., pp.139-156, May 1973.
- [7] Bigelow, R., 'Structured Code via Stack Machine', Computer, p. 67, June 1975.
- [8] Bobeck, A.H., et.al., 'Magnetic Bubbles - An Emerging New Memory Technology', Proc. IEEE, Vol.63, pp. 1176-1195, Aug. 1975.
- [9] Burns, R. and Savilt, D., 'Micro-programming, Stack Architecture Ease Minicomputer Programmer's Burden', Electronics, Vol.46, pp. 95-101, Feb. 1973.
- [10] Carlon, C.B., 'The Mechanization of a Pushdown Stack', AFIPS, Vol. 24, pp. 243-250, 1963.
- [11] Chen, T.C. and Tung, C., 'Storage Management Operations in Linked Uniform Shift-Register Loops', IBM J. Res. Develop., Vol.20, pp.123-131, March 1976.
- [12] Chen, R.C., et.al., 'MININET: A Microcomputer Controlled Mininetwork', Proc. IEEE, Vol.64, pp. 988-993, June 1976.

- [13] Chu, W., 'Buffer Behaviour for Poisson Arrivals and Multiple Synchronous Constant Outputs', IEEE Trans. on Computers, C-19, pp. 530-534, June 1970.
- [14] Cohen, M.S. and Chang, H., 'Frontiers of Magnetic Bubble Technology', Proc. IEEE, Vol.63, No.8, pp. 1196-1206, Aug. 1975.
- [15] Conti, C.J., 'Concepts for Buffer Storage', IBM TR 00.1852, Feb. 25, 1969.
- [16] Deo, N., Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1974.
- [17] Dor, N.M., 'Guide to the Length of Buffer Storage Required for Random (Poisson) Input and Constant Output Rates', IEEE Trans., EC-16, pp. 683-684, Oct.1967.
- [18] Doran, R.W., 'Architecture of Stack Machines', in High Level Language Computer Architecture, edited by Chu.Y., Academic Press, 1975.
- [19] Evey, R.J., 'Application of Pushdown Store Machines', AFIPS, Vol.24, pp. 215-228, 1963.
- [20] Feller, W., An Introduction to Probability Theory and its Applications, Vol.1, John Wiley and Sons, Inc., N.Y., 1968.
- [21] Feth, G.C., 'Memories: Smaller, Faster and Cheaper', IEEE Spectrum, pp. 36-43, June 1976.
- [22] Fisherman, G.S., Concepts and Methods in Discrete Event Digital Simulation, John Wiley and Sons, N.Y., 1973.
- [23] Fuller, S.H., Analysis of Drum and Disk Storage Units, Springer-Verlag, Berlin, 1975.
- [24] Fuller, S.H. and Baskett, F., 'An Analysis of Drum Storage Units', J. ACM, Vol.22, pp. 83-105, Jan. 1975.
- [25] General Purpose Systems Simulator (GPSS)-III, IBM User's Manual, Form H20-0163-1.
- [26] Gluck, S.E., 'Impact of Scratchpads in Design; Multi-Functional Scratchpad Memories in the Burroughs B8500', AFIPS, Vol.27, pp. 661-666, FJCC 1965.
- [27] Halatsis, C., Maritsas, D. and Philokyprou, G., 'Analysis of Block-Organized Buffers for Poisson Input and Periodic Block Input', Int.J.Systems Sci., Vol.5, No.1, pp.29-40, 1974.

- [28] Hahn, J., 'The Design of Buffer Memories for Multi-channel Neutron Time-of-Flight Analysis', IEEE Trans. Nucl.Sc., NS-15, pp. 157-162, Feb. 1968.
- [29] Hauck, E.A. and Dent, B.A., 'B-6500/7500 Stack Mechanism', AFIPS, Vol.32, pp. 245-252, 1968.
- [30] Hoff, M.E. Jr. and Mazor, S., 'Operation and Application of MOS Shift-Registers', Computer Design, Vol.10, pp. 57-62, 1971.
- [31] Juliussen, J.E., 'Magnetic Bubble Systems Approach Practical Use', Computer Design, pp. 81-91, Oct. 1976.
- [32] Kartashev, S.I., 'A Microcomputer with a Shift-Register Memory', IEEE Trans. on Comput. C-25, pp.470-484, May 1976.
- [33] Kleinrock, L., Queueing Systems, Vol.1: Theory, John Wiley and Sons, N.Y., 1975.
- [34] Lee, C.Y. and Paull, M.C., 'A Content Addressable Distributed Logic Memory with Applications to Information Retrieval', Proc. IEEE, Vol.51, pp. 924-932, June 1963.
- [35] Maritsas, D.G. and Hartley, M.G., 'Buffer Length for Erlang Input with Constant Removal Rates', IEEE Trans. on Computers, pp. 839-843, Sept.1970.
- [36] Om Vikas and Rajaraman, V., 'Buffered Electronic Cyclic Memories', Proc. Symposium on Computer Architecture and System Design at I.I.T. Delhi, Nov.24-26, pp. 161-170, 1976.
- [37] Organick, E.I., Computer System Organization, B-5700/6700 Series, ACM Monogram, Academic Press, 1973.
- [38] Panigrahi, G., 'Charge-Coupled Memories for Computer Systems', Computer, pp. 33-42, April 1976.
- [39] Parhami, B., 'Associative Memories and Processors: An Overview and Selected Bibliography', Proceeding of IEEE, Vol.61, No.6, pp.722-730, June 1973.

- [40] Philokyprou, G. and Halatsis, C., 'Electronic Buffer Memories', Int.J. Elect.Engg.Educ., Vol.12, pp. 25-35, 1975.
- [41] Philokyprou, G. and Maritsas, D.G., 'A Class of Buffer Systems with Heterogeneous Input and Output Processes', Int.J.Systems Sc., Vol.5, pp.201-212, 1974.
- [42] Philokyprou, G. and Tzafestas, S., 'Buffer-size and Waiting Time Calculations for Random Arrivals and Periodically Regular Service', Electronics Letters, pp. 588-590, 3rd Sept. 1970.
- [43] Philokyprou, G. and Zacharakopoulos, A., 'The Stacking Register - A Simultaneous Input/Output Buffer', Nucl.Inst.and Methods, Vol.65, pp.202-204, 1968.
- [44] Powell, B. and Avi-Itzhak, B., 'Queueing Systems with Enforced Idle Time', Operations Res., Vol.15, pp.1145-1156, 1967.
- [45] Rajaraman V. and Om Vikas, 'Buffered Stack Memory Organization', Electronics Letters, Vol.11, pp. 305-307, 10th July 1975.
- [46] Salzer, J.M., 'Bubble-Memories - Where Do We Stand?', Computer, pp. 36-41, March 1976.
- [47] Scarrott, G.G., 'The Efficient Use of Multilevel Storage', Proc. 1965 IFIP Congress, Vol.1, Spartan, 1965, pp. 137-141.
- [48] Skinner, C.E., 'Priority Queueing Systems with Server-Walking Time', Operations Res., Vol.15, No.2, pp. 278-285, 1967.
- [49] Slotnick, D.L., 'Logic-Per-Track Devices', Advances in COMPUTERS, Vol.10, (Academic Press, N.Y., 1970), pp. 291-296.
- [50] Terman, L.M. and Heller, L.G., 'Overview of CCD Memory', IEEE Trans. on Electron Dev., ED-23, pp. 72-78, Feb. 1976.

- [51] Torrero, E.A., Microprocessors: New Directions for Designers, Hayden Book Co., 1975.
- [52] Tung, C, Chen, T.C. and Chang, H., 'Bubble Ladder for Information Processing', IEEE Trans. on Magnetism, MAG-11, pp. 1163-1165, Sept. 1975.
- [53] Tzafestas, S. and Philokyprou, G., 'On the Design of Electronic Buffer Memories', Proc. Intl. Symp. on Design and Appl. of Logical Systems, Brussels, pp. 1061-1075, Sept. 1969.
- [54] Wagner, H.M., Principles of Operations Research, Prentice-Hall of India, New Delhi, 1973, pp. 843-846.
- [55] Wensley, J.H., 'The Impact of Electronic Disks on System Architecture', Computer, pp. 44-48, Feb. 1975.
- [56] Wilkes, M.V., 'Slave Memories and Dynamic Storage Allocation', IEEE Trans. on Electron. Comput., EC-14, pp. 270-271, April 1965.
- [57] Williams, J.G., 'Asymmetric Memory Hierarchy', Commun. ACM, Vol.16, pp. 213-222, April 1973.
- [58] Wishart, D.M.G., 'Queueing Systems in which Discipline Is Last-Come-First-Served', Operations Res., Vol.8, pp. 591-599, 1960.

Special Issues on ECMS:

- [i] Large Capacity Digital Storage Systems,
Proc. IEEE, Vol.64, Aug. 1975.
- [ii] Electronic Disks,
Computer, Feb. 1975,
Computer, March 1976,
Computer, April 1976.
- [iii] CCD Memories,
IEEE Journal of Solid-State Circuits, SC-11,
No.1, Feb. 1976.
IEEE Trans. on Electron. Devices, ED-23,
Feb. 1976.
- [iv] Bubble Memories,
IEEE Trans. on Magnetism, MAG-12,
June 1976.

APPENDIX A

We have proposed the schemes for the buffered constant clock ECM organizations with FIFO access discipline in Chapter 2 and with LIFO access discipline in Chapter 5, and the buffered variable clock ECM organization with LIFO access discipline in Chapter 3. The Scheme for multiple insertions and deletions in a buffered ECM has been discussed in Chapter 4. In this appendix, we use the Computer Design Language (CDL) - a highly descriptive language [Y. Chu, 1972], to describe the above schemes.

We use the terms SR and ECM interchangeably.

A-1 BUFFERED SERIAL MEMORY WITH CONSTANT CLOCK ECM

A scheme is proposed in Chapter 2 to realise a serial memory with constant clock ECMs. Addition of buffers on the input and output ends cater to the random insertion and retrieval requirements. A CDL-description of the proposed scheme is given as below.

Comment, Configuration of a buffered serial memory

Register, C1(0-9),	: Tail-distance counter
C2(0-9),	: Head-distance counter
SRC(0-10),	: Shift-register content counter
IBR(0-7),	: Input buffer register
OBR(0-7),	: Output buffer register
I(0-3), J(0-3)	: Switch index pointers
F(0-2),	: Operation register
RI ,	: Insertion request flipflop
RR ,	: Retrieval request flipflop

Array-Register,	B1(1-8,0-7),	: Input Buffer
	B2(1-8,0-7),	: Output Buffer
Array-shift- register,	SR(0-1023,0-7)	: Shift-registers in parallel
Cas-register,	CB = B1(8)...B1(1)-B2(1)...B2(8)	
Cas-shift-regis- ter,	CSR1 = B1(8)...B1(1)-SR(0-1023)	
	CSR2 = SR(0-1023)-B2(1-8)	
Decoder,	K(1-4) = F	
Terminal,	PATH1 = K(1),	
	PATH2 = K(2),	
	PATH3 = K(3),	
	RECL = K(4),	
Switch,	START(ON)	
Clock	P(1-2)	: Clocks for data transfer between buffers
	CLK,	: Clock for shift-register.
Comment,	Start operation	
/START(ON)/	SRC ← 0, I ← 0, J ← 0, C1 ← 0, C2 ← 0, F ← 1	
Comment,	Request for insertion sets RI	
/RI * P(1) /	IF(I ≠ 8) THEN I ← inc I	
	ELSE RI ← 0;	
/RI * P(2)/	B1(I) ← IBR, RI ← 0;	
Comment,	Request for retrieval sets RR	
/RR * P(1)/	IF(J ≠ 0) THEN OBR ← B2(J)	
	ELSE RR ← 0;	
/RR * P(2)/	J ← dec J, RR ← 0;	

```

Comment,          Path1 is activated when SR is empty, B1→B2
/K(1)*P(1)/       IF(J=8)THEN F ←2;
/K(1)*P(2)/       CB ← shr CB
                  I ← dec I, J ← inc J;

Comment,          Activate, Path2 when tail reaches input port
/K(2) * CLK/      IF((I=0)+(SRC=1023)) THEN F ← 4
                  ELSE   CSR1 ← shr CSR1,
                  I ← dec I, SRC ←inc SRC, C2 ←inc C2;

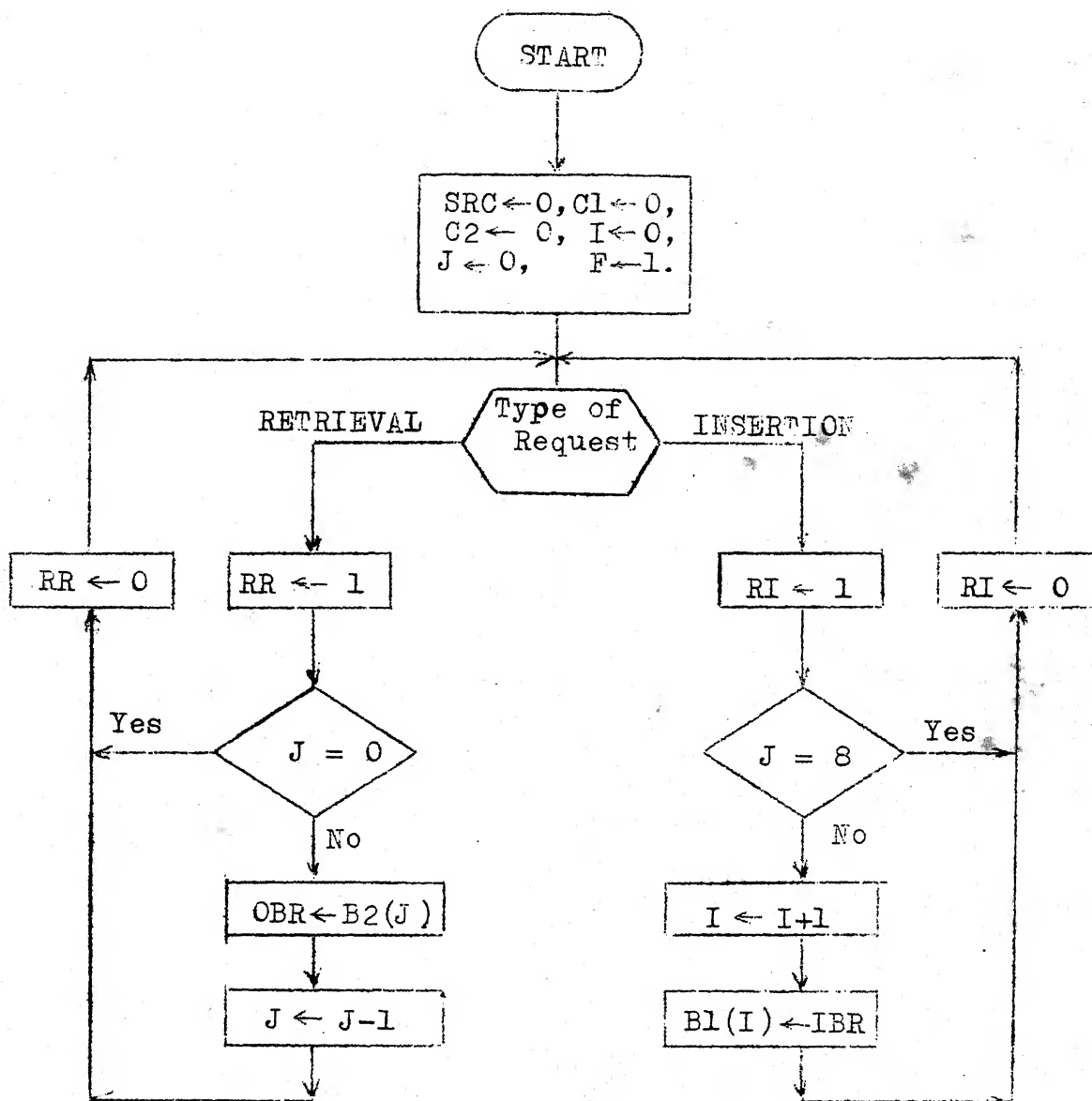
Comment,          Activate Path3 when head reaches output port
/K(3) * CLK/      IF(J=8) THEN F ← 4  ELSE
                  (IF(SRC=0) THEN F ←1 ELSE
                  (CSR2 ← shr CSR2, J ←inc J,
                  SRC ← dec SRC, C1 ←inc C1));

Comment,          Rest of information recirculates
/K(4)*CLK/        SR ← cir SR,
                  C1 ← inc C1,
                  C2 ← Inc C2,
                  Z ← 1,

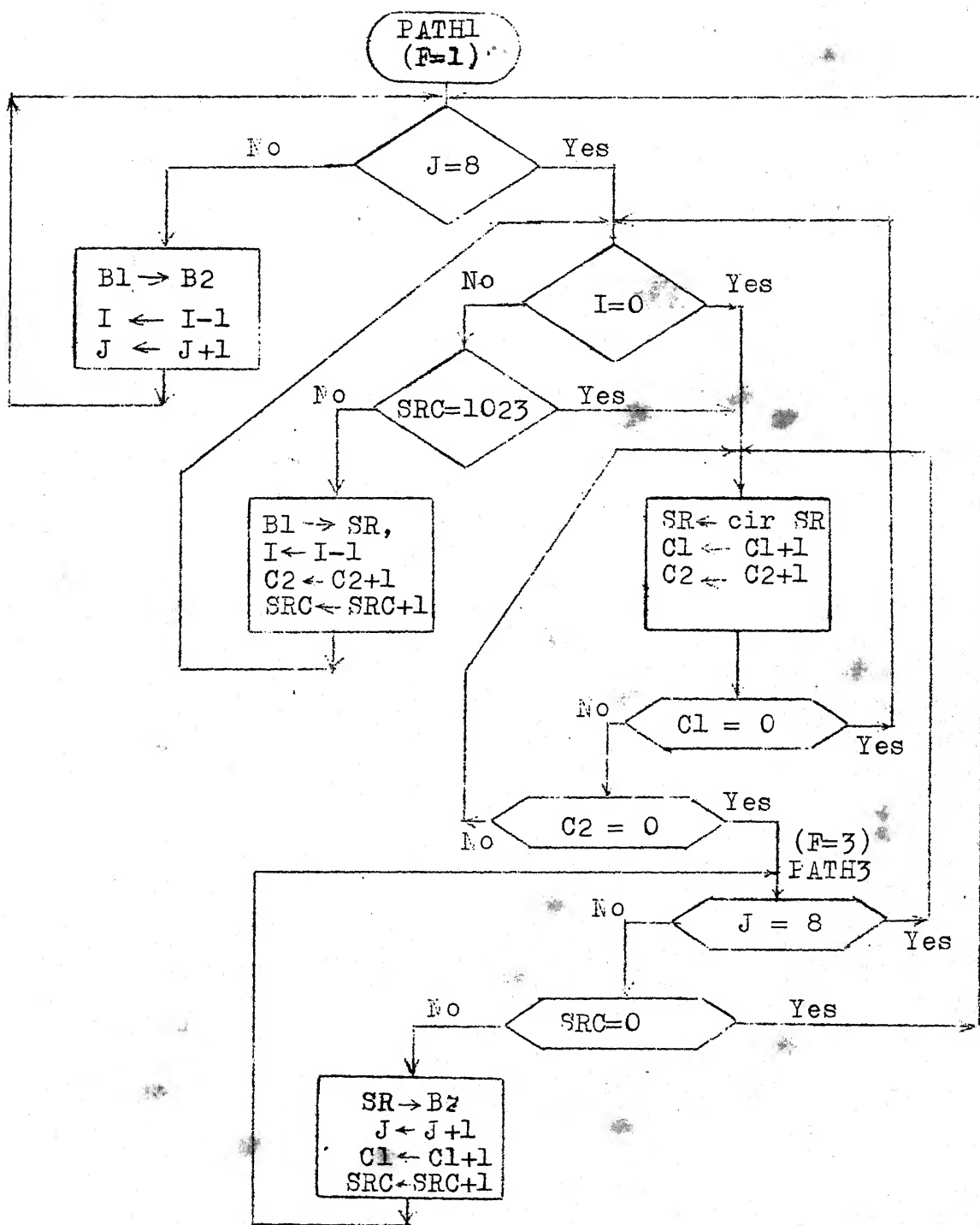
/K(4)*Z*P(1)/     IF(C1=0) THEN F ←2 ELSE
                  (IF(C2=0) THEN F ←3),
                  Z ← 0 ;

```

END



(Continued)



FLOWCHART FOR THE BUFFERED SERIAL MEMORY.

A-2 BUFFERED STACK MEMORY WITH VARIABLE CLOCK ECM

A scheme, for buffered stack memory is proposed in Chapter 3. The stack memory consists of large size static shift-registers and a small size fast stack. A CDL-description of the proposed scheme is given as below.

Comment,	Configuration of a buffered stack memory organization	
Register,	BR(0-15),	: Input buffer register
	S(0-15),	: Register with SS
	SRC(0-11),	: SR-stack content counter
	FSC(0-3),	: FS-content counter
	G(0-1),	: Operation register
	F(0-1),	: FS-status register
	FLAG,	: Initial insertion indicator
	AVL,	: SS-availability indicator
	RI,	: Request for insertion
	RR,	: Request for retrieval
Array-register,	FS(0-7 ,0-15),	: Fast stack
Array- shift- register,	SS(0-1023, 0-15),	: Shift-register stack
Cas-shift- register,	CSS(0-1024) = S-SS(0-1023),	
Cas-register,	CFS = FS-BR	
Comment,	Control Structure	
Switch,	START(ON),	: Start operation
	STOP(ON),	: Stop operation

Decoder, K(0-3) = G

157

C(1-3) = F

Terminal, SS(TOP) = SS(1023,0-15), : Top of SS
FS(TOP) = FS(7,0-15), : Top of FS
FS(BOT) = FS(0, 0-15), : Bottom of FS
FIND = K(0), : Find Operation
DONE = K(1), : Operation done
POPUP= K(2), : Retrieve a word
PUSH = K(3), : Insert a word
FSMT = C(1), : FS is empty
FSPT = C(2), : FS is partially
filled
FSFL = C(3), : FS is full

Clock, CLK,

Comment, Start Operation

/START(ON)/ FSC ← 0, AVL ← 1, G ← 1, SKIP ← 0, TRANS ← 0,
REORG ← 0;

Comment, Find a request

/K(0)* RR/ G ← 2,
IF(FSC=0) THEN F ← 1;

/K(0)* RI/ G ← 3,
IF (FSC=0) THEN F ← 1 ELSE
(IF (FSC=8) THEN F ← 3 ELSE F ← 2);

/K(1)/ RR ← 0, RI ← 0, G ← 0;

Comment, Request for retrieval and FS is empty

/K(2)*C(1)*AVL/ AVL ← 0,
IF(FLAG=1) THEN (SKIP ← 1)
ELSE(TRANS ← 1);

```

/SKIP* CLK/      CSS ← shr CSS,
                  SRC ← dec SRC,
                  FLAG ← 0, SKIP ← 0;

/TRANS* CLK/      IF(SRC=0) THEN GO TO ERR ELSE
                  (BR(0-15) ← SS(TOP),
                  SS ← shr ← SS,
                  G ← 1, AVL ← 1, TRANS ← 0);

Comment,          Request for Retrieval and FS is nonempty.
                  FS → BR.

/K(2)* C(1)/      CFS ← shr CFS,
                  FSC ← dec FSC, G ← 1;

Comment, Request for Insertion and FS is empty.

/K(3)*C(1)*AVL/   S ← BR,
                  CFS ← shl CFS,
                  FSC ← inc FSC,
                  FLAG ← 1, G ← 1, REORG ← 1;

Comment, Reorganize Shift-register stack memory

/REORG*CLK/       IF(SRC = 1024) THEN GO TO ERR ELSE
                  CSS ← recl CSS;

Comment, Request for insertion. FS is found partially
filled. BR → FS

/K(3)* C(2)/      CFS ← shl CFS,
                  FSC ← inc FSC,
                  G ← 1;

```

Comment, Request for insertion. FS is found full.

Skip on FLAG.

/K(3)*C(3)*FLAG/ CFS ← shl CFS,
FLAG ← 0, G ← 1;

Comment, Push into SS when FLAG is reset. BR → FS.

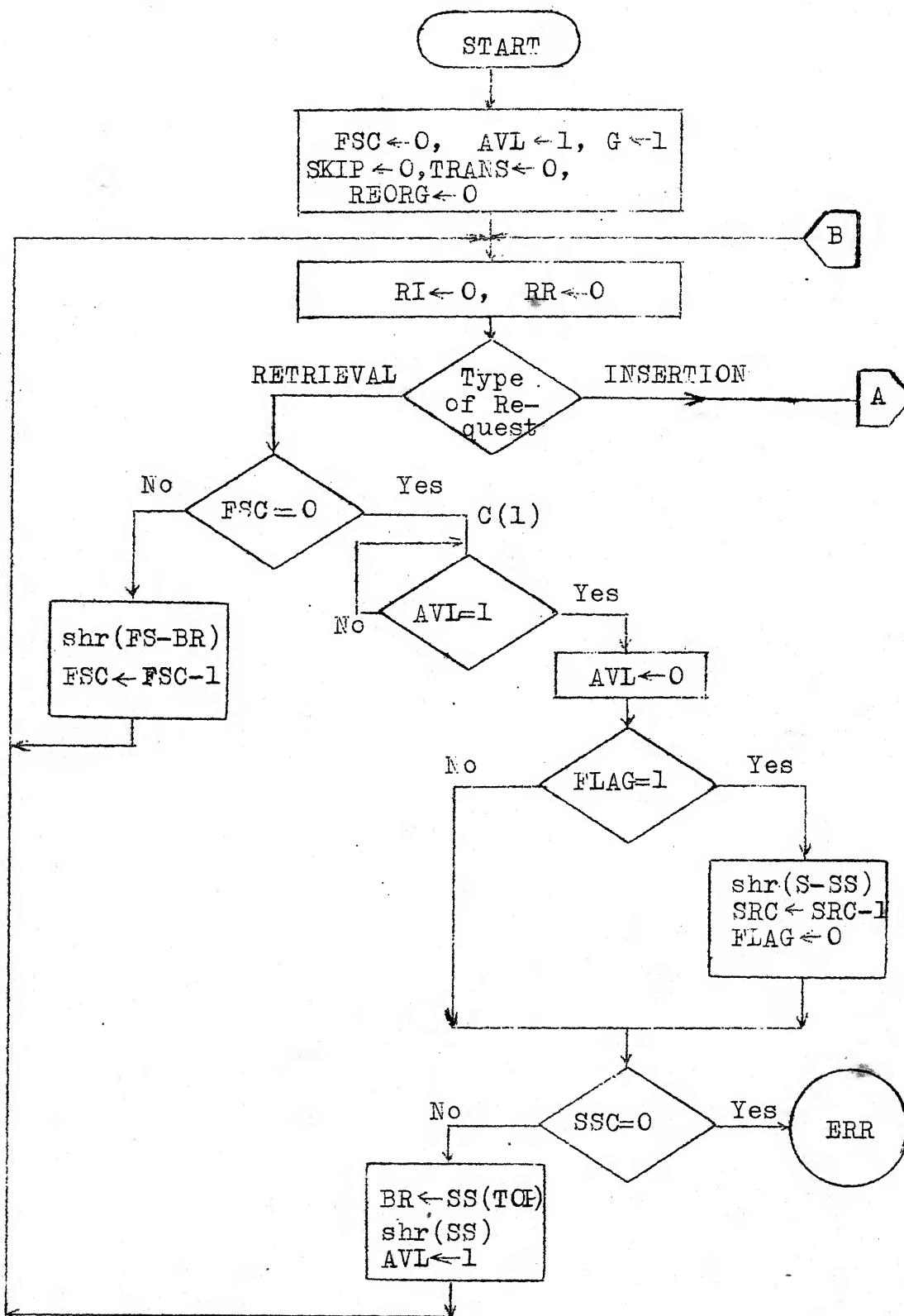
/K(3)*C(3)*AVL*FLAG/ S ← FS(BOT),
CFS ← shl CFS, REORG ← 1, G ← 1;
/ERR/ STOP(ON)

END

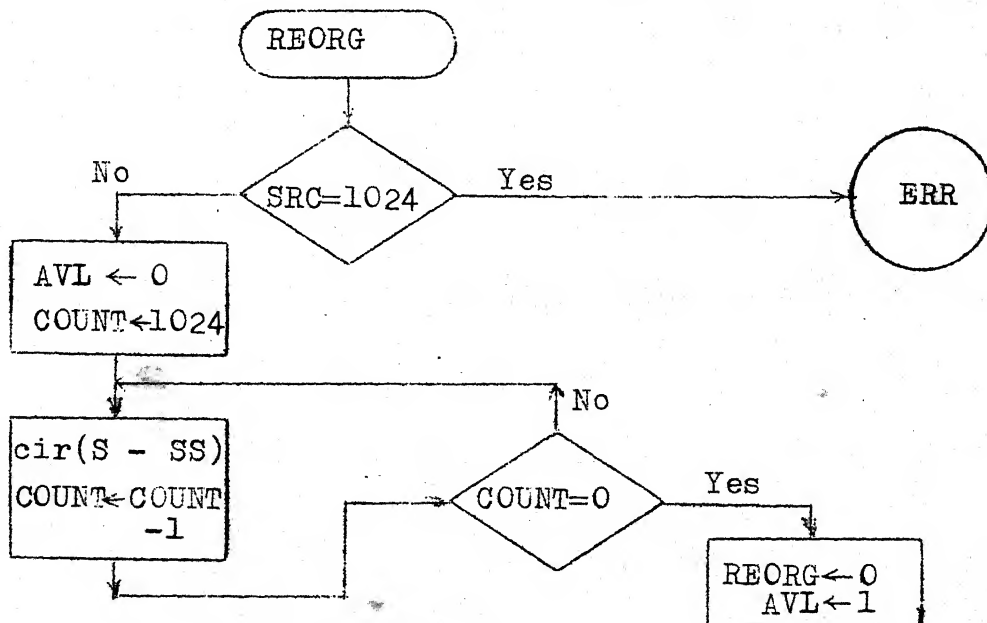
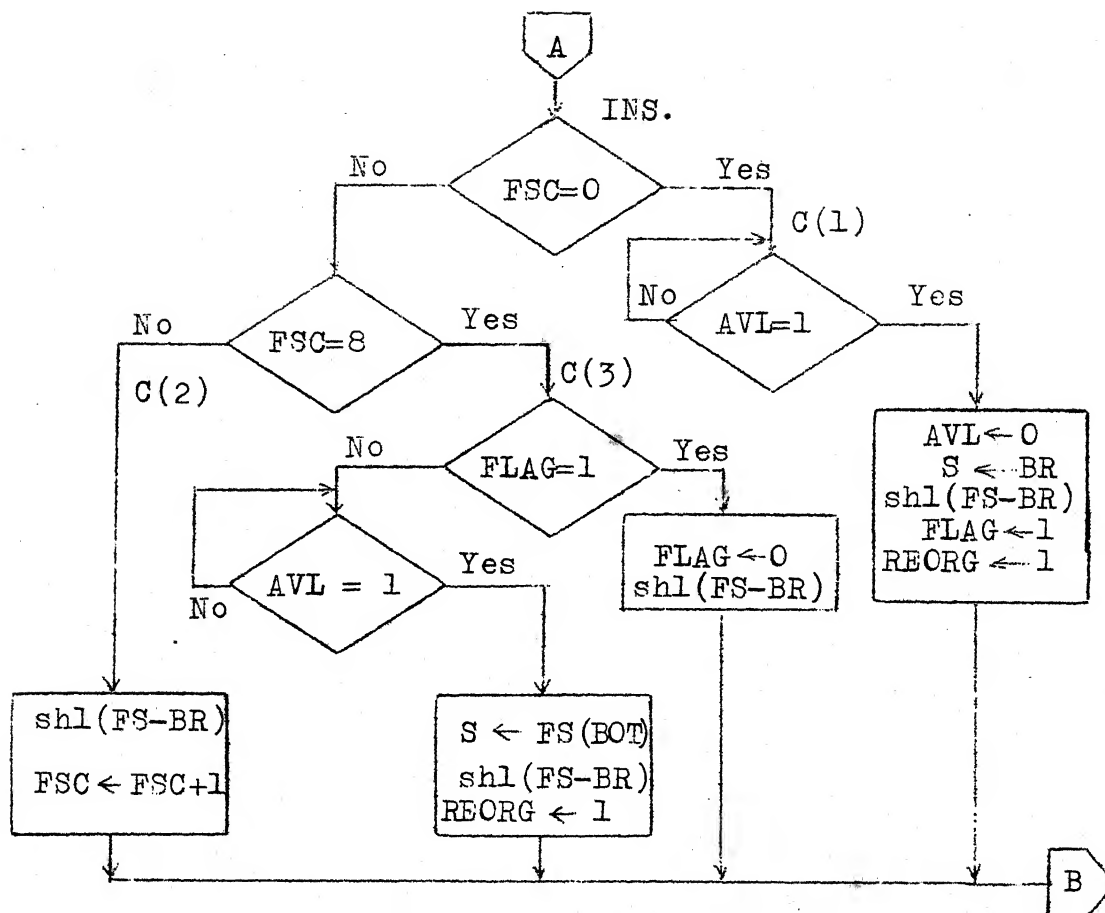
Comment, a special operator to shift circularly
1024 times

Operator CSS ← recl CSS
/BEGIN/ COUNT = 1024, AVL ← 0
/REORG * CLK/ CSS ← cir CSS,
COUNT ← dec COUNT
IF(COUNT = 0) THEN (REORG ← 0,
AVL ← 1)

end of operator.



(Continued)



FLOWCHART FOR THE BUFFERED STACK MEMORY WITH
VARIABLE-CLOCK ECM.

A-3 A SCHEME FOR MULTIPLE INSERTIONS AND DELETIONS IN AN ECM

A scheme for insertion and deletion of a string in an ECM is discussed in Chapter 4. CDL-description of the same is given as below. Extreme states, namely empty and full states, of SR are not considered.

Comment, Configuration of buffered electronic cyclic memory. We consider that maximum 7 words can be either inserted or deleted by a single operation.

Register,	OP(0,2),	: Operation Register
	WC(0,2),	: Word count
	CR(0-7),	: Comparand register
	MASK(0-7),	: Mask Register
	LC(0-9),	: Location counter
	RI,	: Request for insertion
	RD,	: Request for deletion
	MATCH,	: Match indicator
	UNS,	: Unsuccessful search over entire memory
	X,Y,	: Flip-flops
Array register, M(0-7,0-8)		: Memory to store words to be inserted
	BUF(1-8,0-8),	: Buffer registers
Shift-register, SR(0-1023 ,0-8)		

Cas-shift register, $CSR(I) = SR(0-1023)-BUF(1-I)$

$CSR(0) = SR(0-1023)$

Decoder, $K(0-4) = OP$

Terminal, $ACTIV = SR(1023,0)$

Switch, $START(ON)$

$STOP(ON)$

Clock, $P(1-2)$, : Frequencies of $P(1)$ and
 CLK , $P(2)$ are more than 8 times
that of CLK .

Comment, Start operation

$/START(ON)/$ $OP \leftarrow 0, UNS \leftarrow 0, B(,0) \leftarrow 0, GO \leftarrow 1, X \leftarrow 0, Y \leftarrow 0;$

$/K(0)*(RD+RI)/$ $OP \leftarrow 1, LC \leftarrow 0, RI \leftarrow 0, RD \leftarrow 0;$

Comment, Search for a given argument in CR

$/K(1)*CLK/$ $IF(LC=1024) THEN (OP \leftarrow 0, UNS \leftarrow 1)$

$ELSE X \leftarrow 1;$

$/K(1)*X*P(1)/$ $MATCH \leftarrow (SR(1023,1) \oplus CR(0) + \overline{MASK(0)})$

$*(SR(1023,2) \oplus CR(1) + \overline{MASK(1)})$

.

.

$*(SR(1023,8) \oplus CR(7) + \overline{MASK(7)}),$

$X \leftarrow 0;$

$/K(1)*Y*(P(2)/$ $IF(MATCH = 1) THEN (OP \leftarrow 2),$

$Y \leftarrow 0;$

Comment, Operation to perform

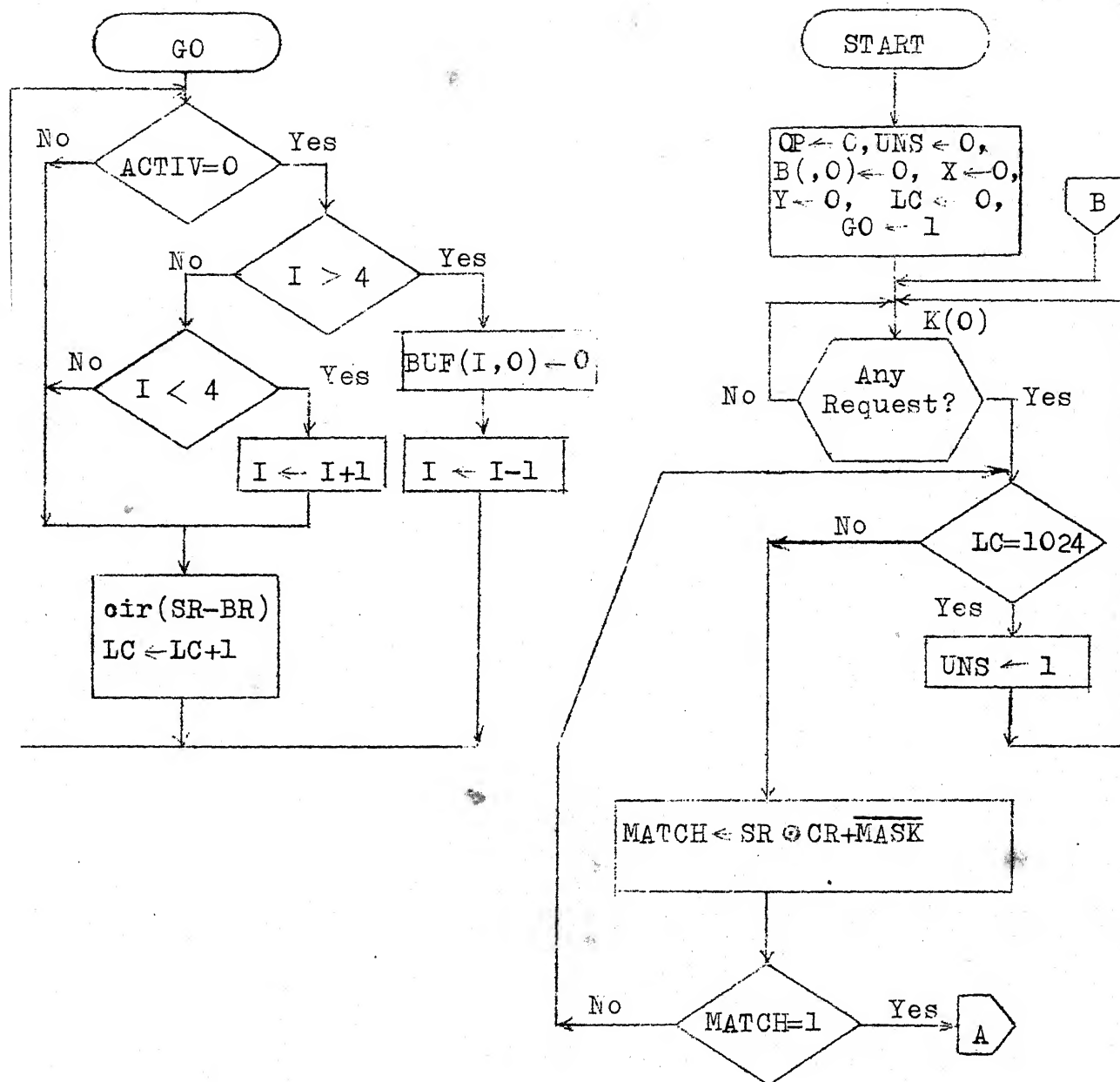
$/K(2)*RD/$ $OP \leftarrow 3, GO \leftarrow 0;$

$/K(2)*RI/$ $OP \leftarrow 4;$

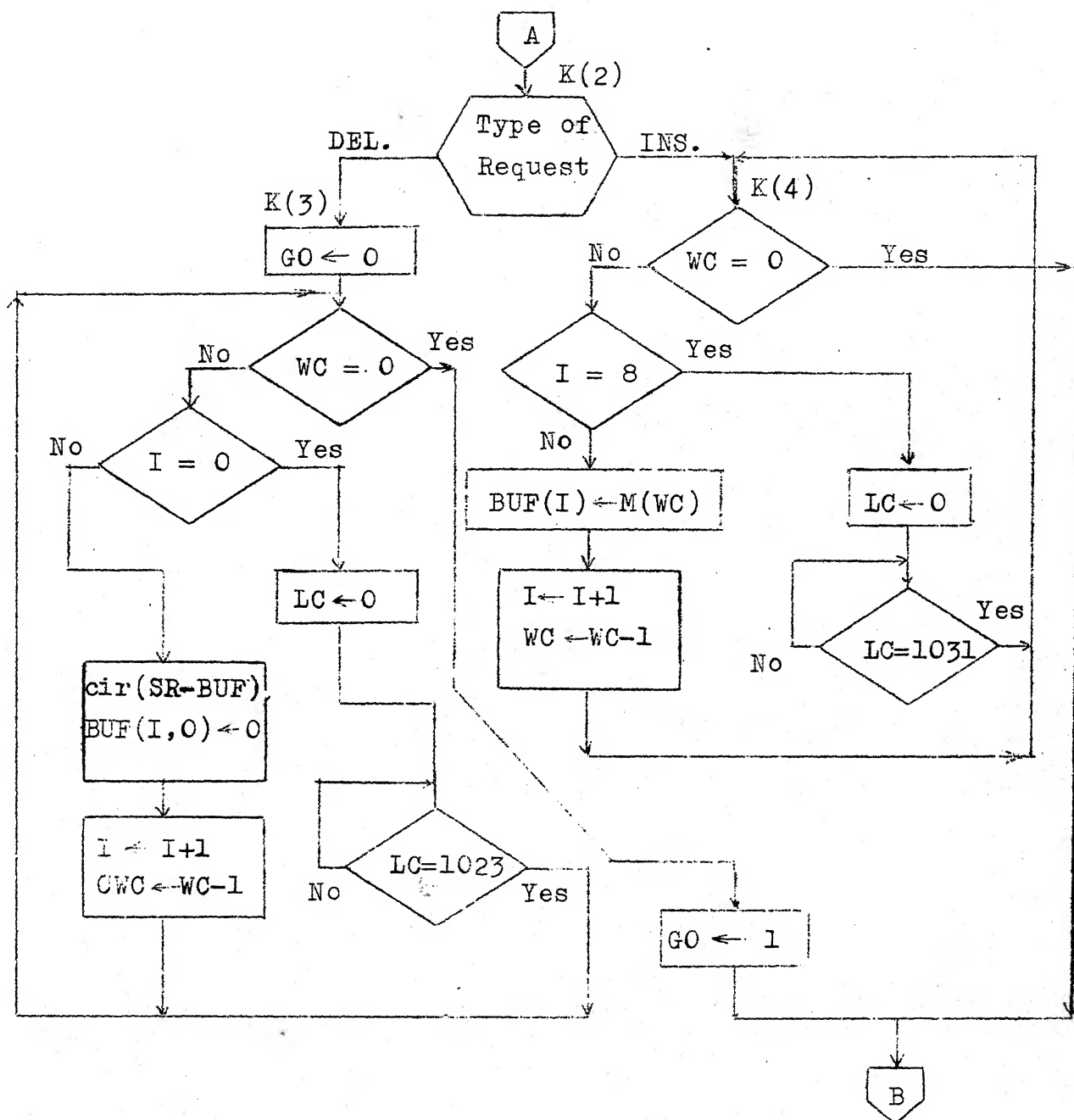
```

Comment,          Deletion of a string
/K(3)* CLK/      IF(WC=0)THEN (GO← 1, OP← 0)
                  ELSE (IF (I=0) THEN (SRC ←0, OP← 5)
                  ELSE (CSR(I) ← cir CSR(I), X ←1));
/K(3) *X *P(1)/  BUF(I,0)← 0;
/K(3) *X *P(2)/  I← dec I, WC← dec WC, X ←0 ;
Comment,          Insertion of a string
/K(4) * CLK/      IF(WC=0) THEN OP ← 0
                  ELSE (IF (I=8) THEN (LC← 0, OP ← 6)
                  ELSE X ←1);
/K(4) *X *P(1)/  BUF(I) ←M(WC)
/K(4) *X *P(2)/  I ← inc I,
                  WC ← dec WC,
                  X ←0 ;
Comment,          Deletion extends over next cycle
/K(5) * CLK/      IF(LC = 1023) THEN (OP ← 3, GO← 0);
Comment,          Insertion extends over next cycle
/K(6) * CLK/      IF(LC = 1031) THEN OP ← 4
Comment,          Information recirculates within SR
/GO* CLK/         IF (ACTIV = 0) THEN
                  (IF(I > 4) THEN X←1 ELSE
                  (IF(I < 4) THEN I← inc I)),
                  CSR(I) ← Cir CSR(I), LC ← inc LC
/GO *X *P(1)/     BUF(I,0)←0
/GO* X *P(2)/     I ←dec I, X← 0
END

```

(Continued)



FLOWCHART FOR
THE SCHEME FOR MULTIPLE INSERTIONS AND DELETIONS IN AN ECM.

A-4 BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM

In this section a new scheme is proposed to realize a stack memory using dynamic shift-registers and a small size buffer (Chapter 5). Information within SR recirculates at a fixed clock-rate. There are virtually two information stacks, namely, one in FS and another in SR. The information stack in SR keeps moving and can be accessed only when its top reaches the output port of SR. At this movement FS is initialized to half of its capacity. The scheme is discussed in Chapter 5. CDL-description of the same is given as below:

Comment,	Configuration of buffered stack	
	memory with constant clock ECM.	
Register,	C1(0-9),	: Tail distance counter
	C2(0-9),	: Head distance counter
	FSC(0-3),	: FS-content counter
	BR(0-7),	: Buffer register
	I(0-2),	} : Switch index pointers
	J(0-2),	
	Q(0-2),	
	F(0-1),	: Operation on FS
	T(0-2),	: Operations on SS-
	RI,	: Insertion request flipflop
	RR,	: Retrieval request flipflop
	X,Y,	: Intermediate setting flipflops.

Comment,	Store a word in FS. BR FS(TOP)
/KF(3) * P(1)/	CFS ← shl CFS, X ← 1;
/KF(3) * X * P(2)/	IF(I=0) THEN J ← inc J
	ELSE (I ← dec I),
	F ← 1, X ← 0;
Comment,	Retrieve a word from FS. FS(TOP) BR
/KF(2) * P(1)/	CFS ← shr CFS, X ← 1;
/KF(2) * X * P(2)/	IF (J = 0) THEN I ← inc I
	ELSE (J ← dec J),
	F ← 1, X ← 0;
Comment,	Test whether head of information chunk
	arrives at the output port of SR.
/KS(0)/	IF(C2=0) THEN GOTO W ELSE T ← 1;
/W/	IF (C1=0) THEN (IF(J=0) THEN T ← 4 ELSE
	GOTO TRANS) ELSE GOTO Z;
/KS(1) * CLK/	SR ← cir SR,
	C1 ← inc C1, C2 ← inc C2,
	T ← 0;
Comment,	Initialize the fast stack
/Z/	IF(J=0) THEN
	(IF(I=0) THEN T ← 1 ELSE (I ← dec I, T ← 2)
	ELSE GO TO TRANS;
/TRANS/	BO ← B2, C2 ← J, I ← 3;

```

Comment,          Fill B1 from SR
/KS(2) * CLK/      B1(I) ← SR(OUT),
                   SR ← shr SR,
                   C1 ← inc C1, X ← 1;

/KS(2) * X * P(1)/ GOTO W, X ← 0;

Comment,          B0 joins SR-data path
/KS(3) * CLK/      CSR(Q) ← cir CSR(Q),
                   C1 ← inc C1,
                   C2 ← inc C2, X ← 1;

/KS(3) * X * P(1)/ IF(C1=0) THEN GO TO ENDF;
/ENDF/             IF(Q=0) THEN T ← 0 ELSE
                   (IF(C2=0) THEN GOTO OVFL0 ELSE T ← 5);

Comment,          Form a normal data path
/KS(5) * CLK/      CSR(Q) ← cir CSR(Q),
                   Y ← 1;

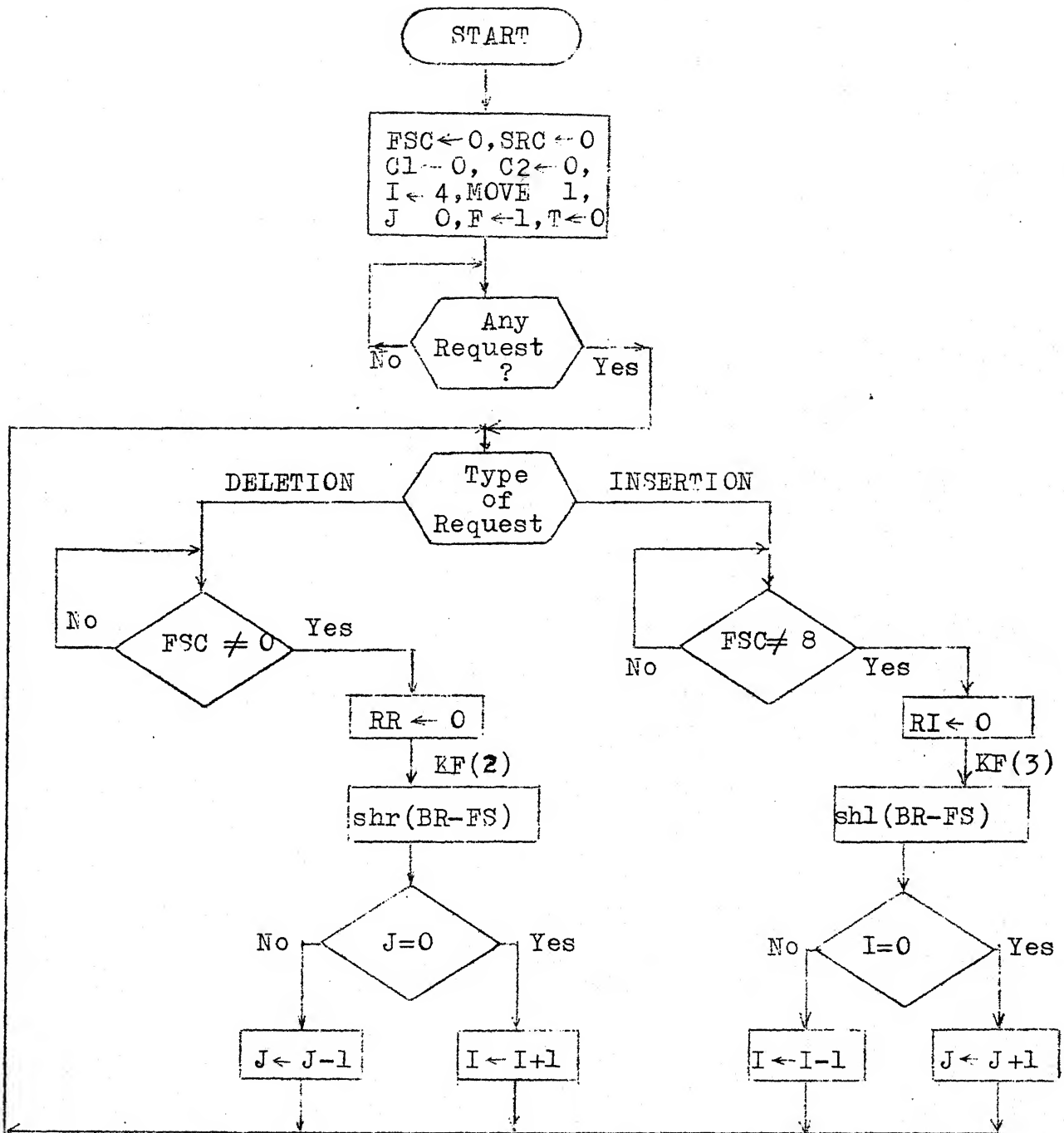
/KS(5) * Y * P(2)/ Q ← dec Q, Y ← 0, GOTO ENDF;

Comment,          Empty SR
/KS(4) * CLK/      SR ← cir SR,
                   IF(J ≠ 0) THEN GOTO TRANS;

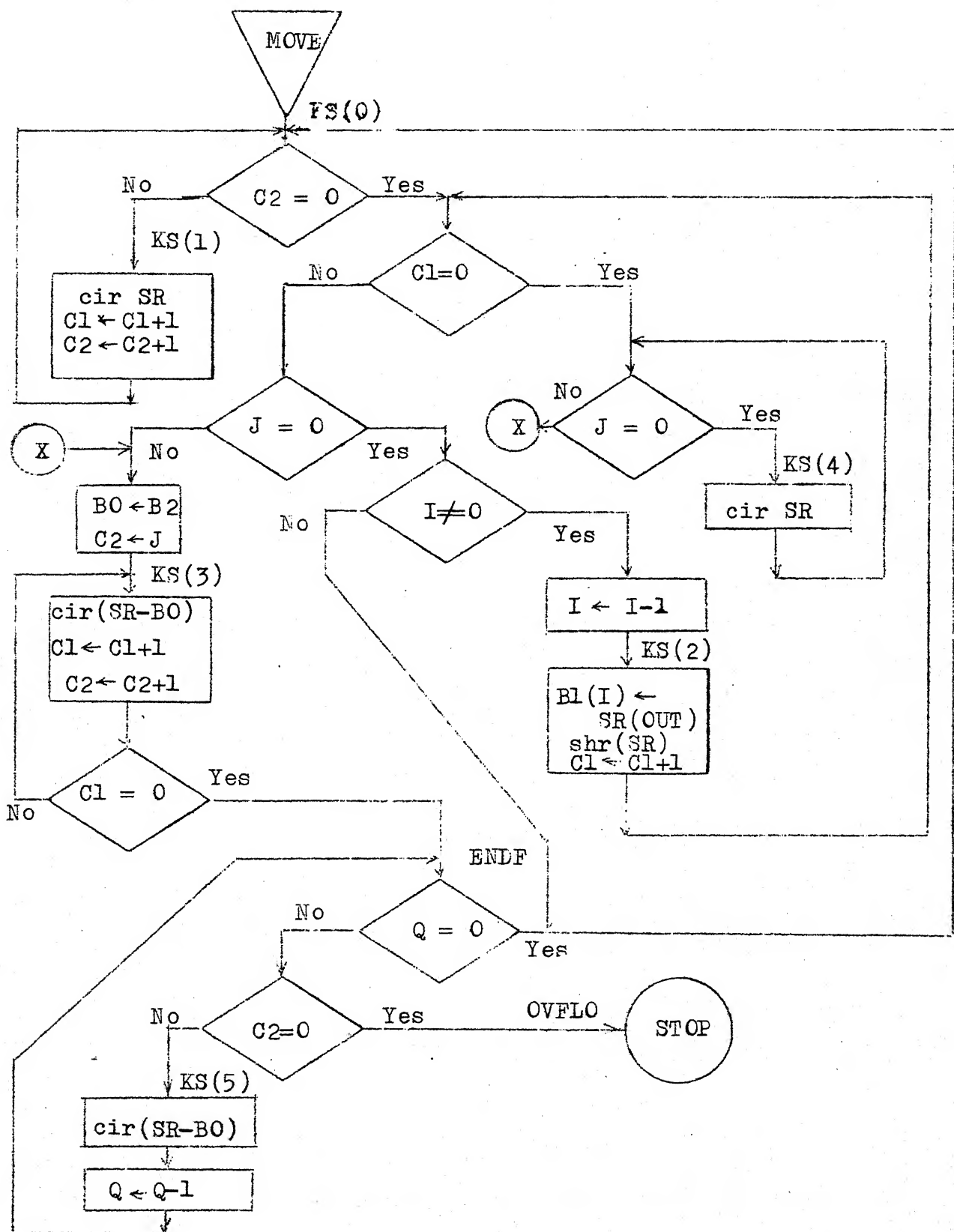
Comment,          SR becomes full before B0 is emptied.
/OVFL0/           STOP(ON)

```

END



(Continued)



FLOW CHART OF A BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM.

APPENDIX B

This appendix contains the listings of the simulation programs, namely M/G/1/L and G/M/1/L finite queue Skinner's models; Buffered Stack Memory with Variable Clock ECM and with Constant Clock ECM; and the scheme for multiple insertions and deletions in an ECM. These programs are written in GPSS III (General Purpose Systems Simulator).

FINITE QUEUE SKINNER'S MODEL

```

*      SIMULATE
*
*      M/G/1/L QUEUEING SYSTEM WITH SERVER RELAXATION TIME
*
1  FUNCTION  RN1,C24
0  0 .1 .104 .2 .222 .3 .355 .4 .509 .5 .69
.6 .915 .7 1.2 .75 1.38 .8 1.6 .84 1.83 .88 2.12
.9 2.3 .92 2.52 .94 2.81 .95 2.99 .96 3.2 .97 3.5
.98 3.9 .99 4.6 .995 5.3 .998 6.2 .999 7 .9998 8
*
1  STORAGE  6          LENGTH OF BUFFER
*
*      ** TIMER **
GENERATE  16000,,,,,1
SAVEX     25+,1
TERMINATE 1
*
*      ** INITIALIZATION **
GENERATE  1,,,1,,1
SIZE2 SAVEX 11,512      SIZE CF SR
MRT SAVEX 12,100
IDRT SAVEX 13,1
POINT SAVEX 20,5000
TERMINATE
*
*      INSERTION
*
GENERATE  X12,FN1
GATE LR  3
LOGIC S  3
SAVEX     1+,1
TEST NE  X1,X20,RESLT
MARK      4
*      GATE SNF 1,LCST      WITH LOSS SYSTEM
GATE SNF  1      WITH WAIT SYSTEM
LOGIC R   3
TABULATE  4
4  TABLE  MP4,64,64,9
ENTER      1
GATE LR    1,DUMP
LOGIC S    1
GATE LR    2
LOGIC S    2
SERVE ADVANCE 0
TABULATE   1
1  TABLE  S1,0,1,18
GATE SNE   1,RELAX
ADVANCE    X13
LEAVE      1
TRANSFER   ,SERVE

```

FINITE QUEUE SKINNER'S MODEL

```
*  
RELAX LOGIC R      1  
      ADVANCE     X11  
      LOGIC R      2  
      TRANSFER     ,DUMP  
  
*  
LOST  SAVEX        7+,1  
DUMP  SAVEX        6+,1  
      TERMINATE  
  
*  
RESLT SAVEX        22,V2  
2      VARIABLE     X7*1000*100/X6  
      TABULATE      4  
      SAVEX         16,V4  
4      VARIABLE     1000-TB4*2000/X11  
      TERMINATE     1000  
  
*  
      START        10000  
      END
```

FINITE QUEUE SKINNER'S MODEL

```

*      SIMULATE
*
*      G/M/1/L QUEUEING SYSTEM WITH SOURCE RELAXATION TIME
*
1      FUNCTION      RN1,C24
0      0      .1      .104 .2      .222 .3      .355 .4      .509 .5      .69
.6      .915 .7      1.2 .75      1.38 .8      1.6      .84      1.83 .88      2.1
.9      2.3 .92      2.52 .94      2.81 .95      2.99 .96      3.2      .97      3.5
.98      3.9 .99      4.6 .995      5.3 .998      6.2 .999      7      .9998 8
*
1      STORAGE      8      SIZE OF BUFFER
*
*      ** TIMER **
GENERATE      16000,,,,,1
SAVEX      25+,1
TERMINATE      1
*
*      ** INITIALIZATION **
GENERATE      1,,,1,,1
SIZE2 SAVEX      11,512
MRT SAVEX      12,100      MEAN INTER REQUEST TIME
IDRT SAVEX      13,1
POINT SAVEX      20,3000      NUMBER OF REQUESTS
ENTER      1,V2
2      VARIABLE      R1/2
TERMINATE
*
*      RETRIEVAL
*
GENERATE      X12,FN1
SEIZE      2
TEST NE      X1,X20,RESLT
SAVEX      1+,1
GATE SNE      1,MISS
ADVANCE      1
LEAVE      1
TRANSFER      ,DRCP
MISS SAVEX      9+,1
DRCP SAVEX      8+,1
RELEASE      2
TERMINATE
RESLT SAVEX      21,V1
1      VARIABLE      X9*1000*100/X8
RELEASE      2
TERMINATE      1000
*
*      INSERTION
*
GENERATE      1
SEIZE      1

```

FINITE QUEUE SKINNER'S MODEL

```
1      TABULATE      1
      TABLE      S1,0,4,33
      GATE SNF      1,RELAX
      ENTER        1
*      TABULATE      2
2      TABLE      S1,0,1,33
      ADVANCE      X13
      TRANSFER      ,FREE
RELAX ADVANCE      X11
FREE  RELEASE      1
      TERMINATE
      START        10000
      END
```

BUFFERED STACK MEMORY WITH VARIABLE CLOCK ECM

* SIMULATE

*-----
* PERFORMANCE EVALUATION OF A STACK HIERARCHY
* STACK CONSISTS OF A FAST STACK AND A LARGE STATIC SHIFT REGISTER
*-----

*
* X1 IS DYNAMIC STORAGE OF FAST STACK AND X2 OF SLOW STACK
* X3 IS MEAN INTER ARRIVAL TIME , X4 CORRESPONDS TO FLAG
* X6 = PRCB (REQUEST IS TO SS FOR RETRIEVAL)
* X7 = PRCB (REQUEST IS TO FS FOR RETRIEVAL OR STORAGE)
* X8 = PRCB (REQUEST IS TO SS FOR INSERTION)
* X9 IS CAPACITY OF FAST STACK AND X10 IS CAPACITY OF SLOW STACK
* X11 FOR WAIT STATE , X12 FOR RUN STATE , X13 FOR BLOCKED STATE
* X14 = REQUEST TO GO FOR SERVICE , X16 = 1000 TIMES THE EFFICIEN
* X17 = 1000 TIMES THE FACILITY UTILIZATION
* X18 = E-COEFFICIENT OF ERLANG DISTRIBUTION
* X19 = LCST ITEMS , X20 = TRANSITION FROM N TO N+1
*

2	FUNCTION	X21,C24
0	0 1000	.104 2000 .222 3000 .355 4000 .509 5000 .69
6000	.915 7000	1.2 7500 1.38 8000 1.6 8400 1.83 8800 2.12
9000	2.3 9200	2.52 9400 2.81 9500 2.99 9600 3.2 9700 3.5
9800	3.9 9900	4.6 9950 5.3 9980 6.2 9990 7 9998 8

* *****
* TIMER
* *****

GENERATE 16000,,,,,1
SAVEVALUE 5+,1 TO TELL NUMBER OF RUNS
TERMINATE 1

* ** INITIALIZATION **

GENERATE	1,,1,,1	
MRT SAVEX	3,100	MEAN INTER REQUEST TIME
SIZE1 SAVEVALUE	9,8	SIZE OF FAST STACK
SIZE2 SAVEVALUE	10,1024	SIZE OF SR STACK
RLANG SAVEX	18,1	ERLANG CONSTANT
POINT SAVEX	30,10000	SAMPLE POINTS
SAVEVALUE	1,V1	FS IS HALF FILLED
SAVEVALUE	2,V2	SS IS HALF FILLED
SAVEX	3,V9	

AHEAD TERMINATE

*
GENERATE 1
GATE NU 3
SEIZE 3
SAVEX 26,X18
SAVEX 21,1000
ERLAN TEST LE 1,X26,BEGIN
SAVEX 21,V7
7 VARIABLE X21*RN2/1000

BUFFERED STACK MEMCRY WITH VARIABLE CLOCK ECM

```

    SAVEX      26-,1
    TRANSFER   ,ERLAN
*
* *****
* GENERATE REQUEST PATTERN
* *****
BEGIN SAVEX      21,V8
8      VARIABLE   9999-X21*10
      ADVANCE    X3,FN2
MOVE   RELEASE    3
      QUEUE      1
      TEST L     Q1,90,RESLT
      GATE NU    1,LCST
      SEIZE      1          ENGAGE FAST STACK
      DEPART     1
      TEST L     X14,X30,RESLT
      SAVEX      14+,1
      MARK       4
      TRANSFER   .5,INSRT,RETRV
*
LOST   SAVEX      19+,1
      DEPART     1
      TERMINATE
*
INSRT  TEST L     X1,X9,FSFUL
      SAVEVALUE  1+,1
      TEST NE    X1,1,FSEMP
OUT    SAVEVALUE  12+,1          RUN STATE
FREE   RELEASE    1
      TABULATE   2
      TERMINATE
FSEMP  SAVEVALUE  4,1
      GATE U     2,IRUN
      SAVEX      13+,1
      TRANSFER   ,RECRG
IRUN   SAVEX      12+,1
      TRANSFER   ,RECRG
BYPAS  SAVEVALUE  4,0
      SAVEX      20+,1
*
* DECREMENT FS BY 1 AND INCREMENT FS BY 1
      TRANSFER   ,FREE
FSFUL  GATE U     2,RUN
      SAVEVALUE  13+,1          BLOCKED STATE
      TRANSFER   ,FLAG
RUN    SAVEVALUE  12+,1
FLAG   TEST NE    X4,1,BYPAS          TEST FLAG WHETHER BOFS IS TOSS
RECRG  SEIZE      2          ENGAGE SLOW STACK
      RELEASE    1
      TABULATE   2

```

BUFFERED STACK MEMCRY WITH VARIABLE CLOCK ECM

TABLE	MP4,0,2,10	SERVICE TIME
START	10000	
END		

BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM

```

*      SIMULATE
*
*      BUFFERED ELECTRONIC CYCLIC MEMORY WITH LIFO ACCESS DISCIPLINE
*
*      -----
*      PERFORMANCE EVALUATION OF STACK MEMORY HIERARCHY
*      STACK CONSISTS OF A LARGE DYNAMIC SR AND A FAST STACK TOP
*      -----
2      FUNCTION      RN2,C24
0      0      .1      .104 .2      .222 .3      .355 .4      .509 .5      .69
.6      .915 .7      1.2 .75      1.38 .8      1.6 .84      1.83 .88      2.1
.9      2.3 .92      2.52 .94      2.81 .95      2.99 .96      3.2 .97      3.5
.98      3.9 .99      4.6 .995      5.3 .998      6.2 .999      7      .9998 8
*
*      ** TIMER **
*      GENERATE      16000,,,,,1
*      SAVEX          25+,1
*      TERMINATE      1
*
*      ** INITIALIZATION **
2      STORAGE      8      LENGTH OF BUFFER
*
*      GENERATE      1,,,1,,1
SIZE1 SAVEX          1,1024      SIZE OF SR
MRT   SAVEX          3,100      MEAN INTER REQUEST TIME
      SAVEX          10,V2      CURRENT CONTENT IN SR
2      VARIABLE      X1/2
RLANG SAVEX          18,1      ERLANG CONSTANT
POINT SAVEX          20,2000
      SAVEX          17,V4
4      VARIABLE      X3/X18
HALF  SAVEX          5,4
      ENTER          2,X5
NUMI  SAVEX          12,2      BURST LENGTH
      TERMINATE
*
*      -----
*      GENERATE REQUESTS PATTERN
*      -----
*      GENERATE      1
*      GATE LR        3
*      TEST LE        X15,X20,RESLT
*      LOGIC S        3
*      SAVEX          26,X18
ERLAN TEST LE        1,X26,BEGIN
*      ADVANCE        X17,FN2
*      SAVEX          26-,1
*      TRANSFER        ,ERLAN
BEGIN ADVANCE        0
BURST SAVEX          13,V1      BURST LENGTH
1      VARIABLE      X12*RN1/1000+1

```


BUFFERED STACK MEMCRY WITH CONSTANT CLOCK ECM

```

      TEST G      X13,0,BURST
      TABULATE    3
3     TABLE      X13,4,4,5
      ASSIGN      3,X13
      SAVEX        15+,1
      TABULATE    2
2     TABLE      S2,0,1,7    OCCUPANCY OF BUFFER
      MARK        1
      LINK         1,FIFO,INCUT
INCUT TRANSFER    .5,INSRT,DELET
INSRT SAVEX       6+,1
      GATE SNF     2
      ENTER        2
      LOCP         3,INSRT
PUT   TABULATE    1
1     TABLE      MP1,64,64,5    WAITING TIME
      UNLINK       1,INCUT,1
SKIP  LOGIC R      3
      TERMINATE
DELET SAVEX       8+,1
      GATE SNE     2
      LEAVE        2
      LOCP         3,DELET
GET   TABULATE    1
      UNLINK       1,INCUT,1
      LOGIC R      3
      TERMINATE
RESLT SAVEX       16,V6    STACK IMPROVEMENT
6     VARIABLE     1000-TB1*2000/X1
      TERMINATE    1000
*
      GENERATE     1
      GATE LR       1
      LOGIC S       1
SERVE TEST NE      S2,X5,RELAX
      TEST L        S2,X5,DRAW
      ENTER         2
      ADVANCE       1
      SAVEX         10-,1
      TEST G        X10,0,HEAD
      TRANSFER      ,SERVE
DRAW  SAVEX        19,S2
      SAVEX         19-,X5
      LEAVE         2,X19
      SAVEX         10+,X19
      TEST LE       X10,X1,HEAD
RELAX ADVANCE      X1
HEAD  LOGIC R      1
      TERMINATE
*

```

BUFFERED STACK MEMORY WITH CONSTANT CLOCK ECM

START
END

10000

MULTIPLE INSERTIONS AND DELETIONS IN ECM

```

*      SIMULATE
*      BUFFERED CYCLIC MEMORY FOR SYMBOL PROCESSING
*
*      BUFFERED ECM FACILITATES MULTIPLE INSERTIONS AND DELETIONS
*
1      FUNCTION      RN1,C24
0      0      .1      .104 .2      .222 .3      .355 .4      .509 .5      .6
.6      .915 .7      1.2      .75      1.38 .8      1.6      .84      1.83 .88      2.1
.9      2.3      .92      2.52 .94      2.81 .95      2.99 .96      3.2      .97      3.5
.98      3.9      .99      4.6      .995      5.3      .998      6.2      .999      7      .9998 8
*
*      ** TIMER **
GENERATE      16000
SAVEX      19+,1
TERMINATE      1
*
*      ** INITIALIZATION **
2      STORAGE      8      LENGTH OF BUFFER
*
GENERATE      1,,1,,1
SIZE1 SAVEX      1,512      SIZE OF SR
MRTT SAVEX      3,100      MEAN INTER REQUEST TIME
NUM1 SAVEX      5,1      MEAN LENGTH OF INS./DEL.
SAVEX      10,V2
2      VARIABLE      X1/2
ENTER      2,V5
5      VARIABLE      R2/2
POINT SAVEX      20,5000      NUMBER OF REQUESTS
TERMINATE
*
*      -----
*      GENERATE REQUESTS PATTERN
*      -----
GENERATE      X3,FN1
TEST L      CH1,1
MARK      8
LINK      1,FIFO,MANIP
MANIP SAVEX      7,V3      LOCATION
MARK      1
3      VARIABLE      X10*RN3/1000
TEST L      X15,X20,RESLT
*
PORT TEST GE      X7,X11,MCVE
SAVEX      12,X7      LOCATION YET TO ARRIVE
SAVEX      12-,X11
ADVANCE      X12      DESIRED LOCATION IS REACHED
SAVEX      11,X7
TABULATE      3
3      TABLE      S2,0,4,18
*
TRANSFER      .5,XPAND,SHORT

```

MULTIPLE INSERTIONS AND DELETIONS IN ECM

```

*
XPAND SAVEX      9,V1      BURST LENGTH
1      VARIABLE  X5*RN2/1000+1
      TEST G     X9,0,XPAND

*
INSRT TEST LE    X9,R2,PUT
      TEST G     X1,X10,FULL
      ENTER      2,X9
      SAVEX      10+,X9
      SAVEX      11+,X9
      SAVEX      15+,1
      TRANSFER   ,FREE

*
SHORT SAVEX      9,V1      BURST LENGTH
      TEST G     X9,0,SHORT

*
DELET TEST LE    X9,S2,OUT
      TEST L     K0,X10,EMPTY
      LEAVE      2,X9
      SAVEX      10-,X9
      ADVANCE     X9
      SAVEX      16+,1

*
FREE  TABULATE   1
1      TABLE    MP8,64,128,5
      TABULATE   2
2      TABLE    MP1,64,128,5
DUMP  UNLINK     1,MANIP,1
      TERMINATE

*
FULL  SAVEX      22+,1
EMPTY SAVEX      22+,1
RESLT SAVEX      16,V7
7      VARIABLE  TB1*1000/X1
      TERMINATE  2000

*
PUT   SAVEX      9-,R2
      SAVEX      10+,R2
      SAVEX      11+,R2
      ENTER      2,R2
      ADVANCE     X1
      ADVANCE     X8
      LEAVE      2,X8
      SAVEX      13+,1
      TRANSFER   ,INSRT

*
MOVE  ADVANCE     V6
6      VARIABLE  X2/2+X1-X11
      SAVEX      11,0      HEAD REACHES PORT
      TRANSFER   ,PCRT

```

MULTIPLE INSERTIONS AND DELETIONS IN ECM

*

```
CUT  SAVEX      9-,S2
      SAVEX     10-,S2
      LEAVE     2,S2
      ADVANCE   X1
      ENTER     2,X8
      ADVANCE   X8
      SAVEX     14+,1
      TRANSFER  ,DELET

      START     100C0
      END
```

Date Slip **A 54007**

[illegible]

EE-1977-D-VIK-DES.